HP Diagnostics Guide

V2500 Server

First Edition

A5075-96006
HP Diagnostics Guide: V2500 Server
Customer Order Number: A5075-90006
December 1998

Printed in: USA

Revision History

Edition: First

Document Number: A5075-90006
Remarks: Initial release. December, 1998.

Notice

The information contained in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Contents

Figures

Tables

Preface

This document describes the offline diagnostics for V2500 servers. It is not intended to be a tutorial or troubleshooting guide but a reference guide that contains information on all utilties and scripts used to troubleshoot these systems.

Notational conventions

This section describes notational conventions used in this book.

`bold monospace`	In command examples, **`bold monospace`** identifies input that must be typed exactly as shown.
`monospace`	In paragraph text, `monospace` identifies command names, system calls, and data structures and types. In command examples, `monospace` identifies command output, including error messages.
italic	In paragraph text, *italic* identifies titles of documents. In command syntax diagrams, *italic* identifies variables that you must provide. The following command example uses brackets to indicate that the variable *output_file* is optional: `command` *input_file* [*output_file*]

Brackets ([ ])	In command examples, square brackets designate optional entries.
Curly brackets ({}), Pipe (|)	In command syntax diagrams, text surrounded by curly brackets indicates a choice. The choices available are shown inside the curly brackets and separated by the pipe sign (|). The following command example indicates that you can enter either a or b: `command {a | b}`
Keycap	**Keycap** indicates the keyboard keys you must press to execute the command example.

NOTE A note highlights important supplemental information.

CAUTION A caution highlights procedures or information necessary to avoid damage to equipment, damage to software, loss of data, or invalid test results.

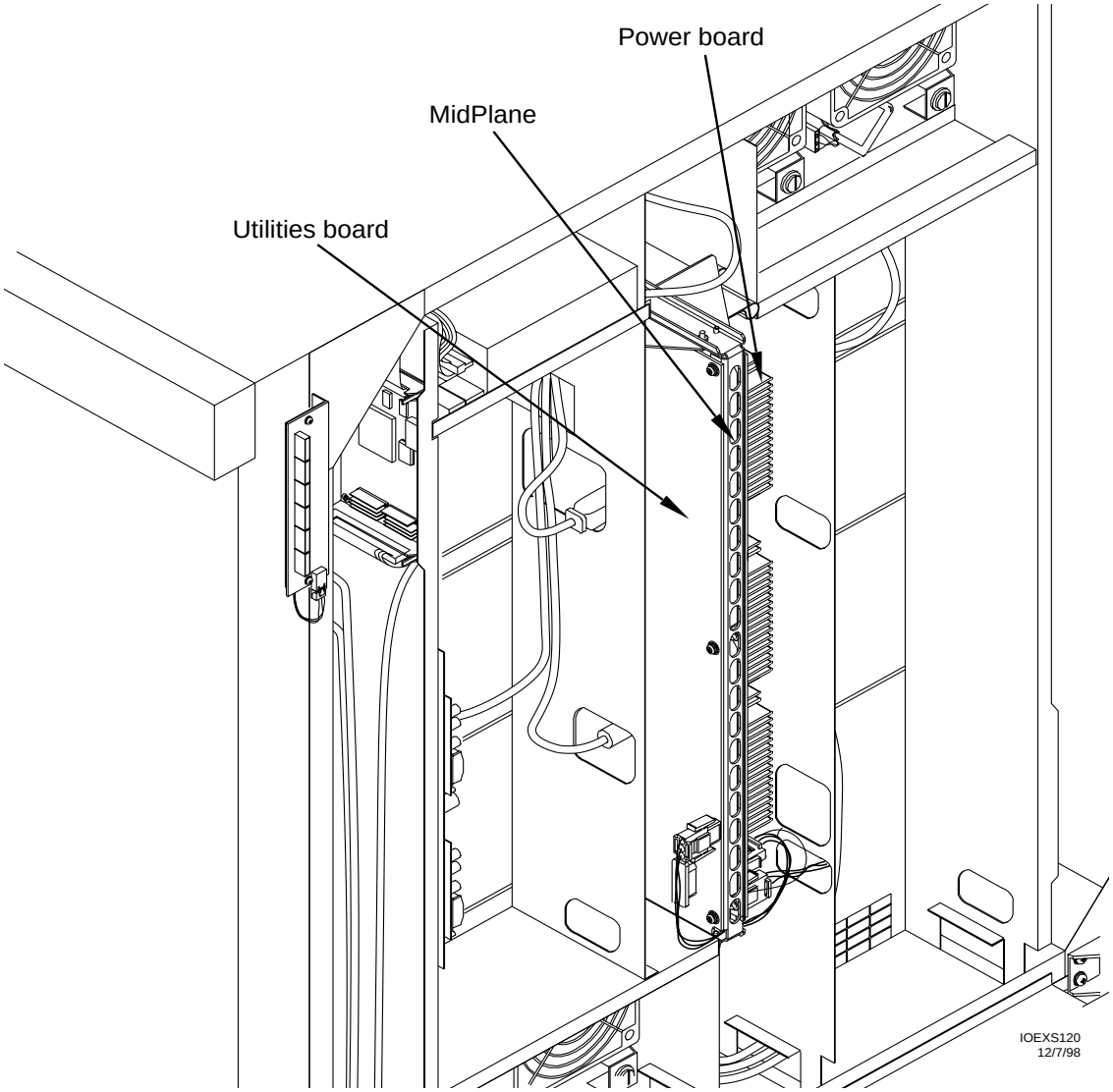
1 Introduction

This chapter presents an overview of the diagnostic mechanism for V2500 servers.

Utilities board

The diagnostic mechanism in the V2500 servers is centered around the Stingray Core Utilities board (SCUB). The SCUB is mounted under the MidPlane Interconnect board (MIB) toward the front of the system. See Figure 1.

Figure 1 Location of the Utilities board



The following devices connect to the Utilities board:

- Core logic bus
- Environmental sensors
- Test points
- Liquid crystal display (LCD)
- Attention lightbar
- Teststation

  The teststation connects to the system via the ethernet and RS232 connections. It is used to configure and run diagnostics on the system. A system will boot and operate without a teststation, and failure of the teststation will not cause interruption of the system.

Figure 2 shows the Utilities board functional layout.

The following hardware components comprise the Utilities board:

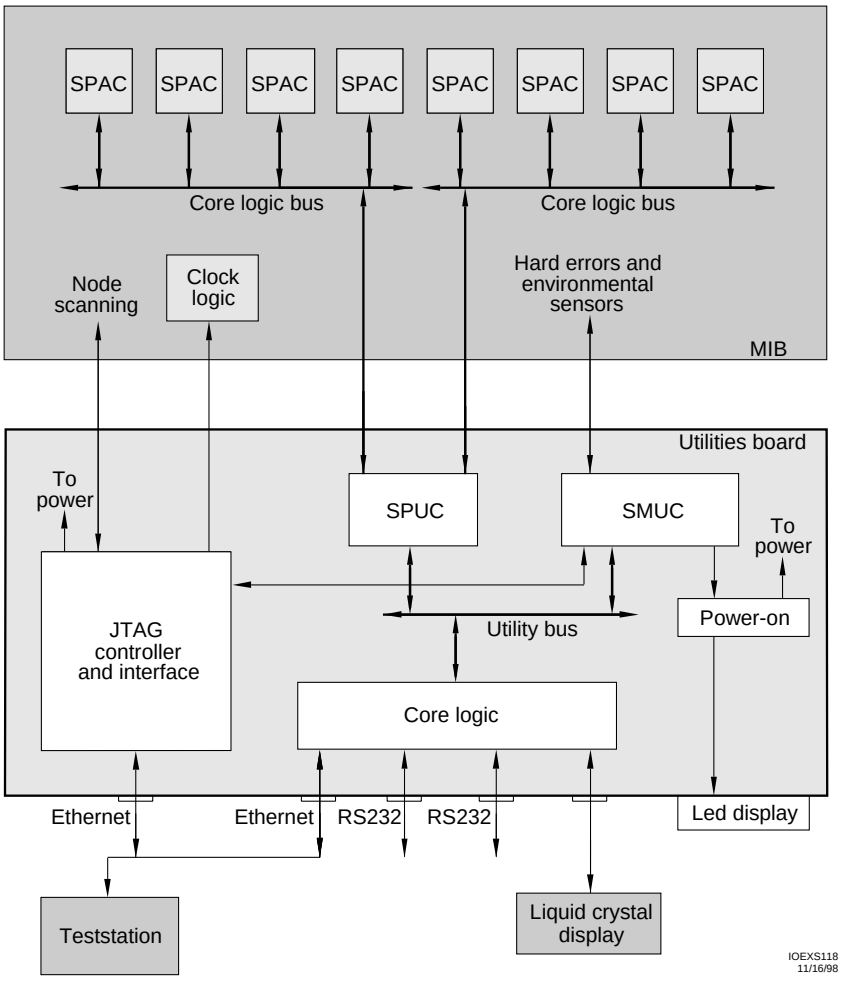
- Core logic—Contains initialization, booting firmware, controller for ethernet and RS-232 interface, and various memories.
- Stingray Monitor Utilities controller (SMUC)—Collects environmental interrupts.
- Power-On circuit—Controls powering up the entire system.

  Environmental sensors are located throughout the system and connect to the SMUC. The SMUC latches interrupts from these sensors as well as other interrupts. The SMUC and the power-on circuit together control system power-up. The power-on circuit drives the attention lightbar diagnostic display through which the operator can determine power-on status.
- Stingray Processor Utilities controller (SPUC)—Interfaces to the core logic bus.

  The SPUC connects to the two core logic buses. Each bus connects up to four Stingray Processor Agent Controllers (SPACs).
- JTAG (Joint Test Action Group) interface—Supports a teststation for running diagnostics. The V2500 servers use a test method called scanning to test boards and other hardware units.

The microprocessor-controlled JTAG interface captures incoming command packets and sends out scan information packets across the ethernet connection to the teststation. Through the teststation connection, one can read and write every CSR in the system.

Figure 2 Utilities board



Core logic

The core logic contains initialization and booting firmware and is described in the following sections.

Flash memory

The core logic contains a four-MByte electrically erasable programmable read only memory (EEPROM) storage for Processor-Dependent Code (PDC). PDC consists of Power-On Self Test (POST) and Open Boot PROM (OBP). The V2500 server uses these two components plus additional firmware called spp_pdc that is laid over OBP and interfaces OBP to HP-UX. Flash memory also contains all diagnostic test, utilities , and scripts.

Flash memory is configured as 512-KByte addresses by 32 data bits with only 32-bit read and write accesses allowed. EEPROM devices are used for flash memory so that it may be rewritten for field upgrades. It can also be written when the SPUC is *scanned*.

Nonvolatile static RAM

The core logic section contains a nonvolatile battery-backed 128-Kbyte RAM (NVRAM) for storing system log and configuration information. This RAM is byte addressable and can be accessed even after power failures.

DUART

A Dual Universal Asynchronous Receiver-Transmitter (DUART) provides to RS232 serial ports and a single parallel port. One serial port provides an interface to a terminal used as a local console to analyze problems, reconfigure the system, and provide other user access.The parallel port of the DUART drives the LCD. The second RS232 port can be used for a modem for field service.

RAM

Random access memory (RAM) provides support for the core system functions. When the system powers up, the processors operate out of this RAM to run self test and configure the rest of the node. Once the system is fully configured, the processors execute out of main memory. The RAM is byte addressable and is 512 KBytes, configured as 128-KByte addresses by 32 data bits.

Console ethernet

The ethernet I/O port provides a connection to the teststation over LAN1.

Attention lightbar and LCD

The attention light bar displays environmental information, such as the source of an environmental error that caused the Utilities board to power down the node.

The liquid crystal display provides basic system information. The core logic drives the LCD through the parallel port on the DUART. The attention lightbar and LCD are detailed in “System displays” on page 12.

COP interface

A serial EEPROM (referred to as COP chip) is located on major boards with information such as serial number, assembly revision, wire revision, truncated board part number, and so on. The SMUC connects to the COP bus selector (CBS) chip on the MIB allowing each COP chip in a node to be read.

SPUC

The SPUC provides interrupts and error messages to and receives control messages from the processors through two 18-bit, bidirectional buses. Each bus connects up to four SPACs. The SPUC also provides core logic bus arbitration for the processors.

SMUC and Power-on

The SMUC registers system environmental parameters. It connects to the utilities bus so that processors can monitor the node by accessing the appropriate CSRs. The SMUC works in conjunction with the power-on circuit to power up the entire system, and it can operate when the rest of the node is powered off or in some indeterminate state. The SMUC drives the environment LCD display. The teststation can also read the environmental LCD display using the `sppdsh` utility. See “sppdsh” on page 268.

SMUC environmental monitoring

The following environmental conditions are monitored:

- ASIC installation error sensing
- FPGA configuration and status
- Thermal sensing
- Fan Sensing
- Power failure sensing
- 48-V failure
- 48-V maintenance
- Ambient air temperature sensing.
- Power-on

Table 1 Environmental conditions monitored by the SMUC and power-on circuit

Condition	Type	Action
ASIC Not Installed OK	Environmental error	Power not turned on, LED indication
FPGA not OK	Environmental error	Power not turned on, LED indication
48-V Fail	Environmental error	Power turned off, LED indication
MIB power fail	Environmental error	Power turned off, LED indication
Board over temp	Environmental error	Power off in one second, LED indication interrupt
Fan not turning	Environmental error	Power off in one second, LED indication interrupt
Ambient air hot	Environmental error	Power off in one second, LED indication interrupt
Other power fail	Environmental error	Power off in one second, LED indication interrupt

Condition	Type	Action
Ambient air warm	Environmental warning	LED indication, interrupt
48-Volt maintenance	Environmental warning	LED indication, interrupt
Hard error	Hard error	LED indication, interrupt

Environmental condition detected by power-on function

The power-on function detects environmental errors (such as ASIC Not Installed OK or FPGA Not OK). It does not turn on power to the node until the conditions are corrected. It also detects environmental errors such as 48-V Fail while the system is powering up and MIB Power Fail after the system has powered up. If a failure is detected in these two cases, the power-on circuit turns off power to the system.

Environmental warnings such as 48-Volt maintenance are also detected by the power-on circuit.

In all cases, the power-on circuit sets an environmental attention light bar code. The code is prioritized so that it displays the highest priority error or warning. See “Attention light bar” on page 16 for a list of codes.

Environmental conditions detected by SMUC

The SMUC detects most of the environmental conditions. It samples error conditions during a time period derived from a local 10-Hz clock that drives the power-on circuit. It registers all the environmental error conditions twice and then logically ORs them together. If the conditions persist for 200 mS, the environmental error bit is set, and an environmental error interrupt is sent to the SPUC, which sends it on to the processors. The SMUC then waits 1.2 seconds and commands the power-on circuit to power down the system.

This same procedure exists for an environmental warning, except that an environmental warning interrupt is sent and the power-on circuit does not power down the system.

The environmental error interrupt and the 1.2 second delay provide the system adequate time to read CSRs to determine the cause of the error, log the condition in NVRAM, and display the condition on the attention lightbar.

After the system is powered down, the Utilities board is still powered up, but all outputs are disconnected from the system.

Environmental control

The Utilities board performs the following functions to control the node environment.

Power-on

When the power switch is turned on, the outputs of the 48-Volt power supplies become active. Several hundred milliseconds after the Utilities board 5-Volt supply reaches its nominal level, the power-on circuit starts powering up the other DC-to-DC converters of the node in succession.

The power-on circuit does not power up the node if an ASIC is installed incorrectly (see “ASIC installation error” on page 18) or if an FPGA is not configured (see “FPGA configuration and status” on page 19). It keeps the system powered up unless an environmental condition occurs that warrants a power-down.

Voltage margining

Voltage margin is divided into four groups called quadrants. The user can margin quadrants separately. When setting the upper margin, for example, all boards in that quadrant are margined for upper.

Clock margining

Parallel ports on the core logic microprocessor select the nominal, upper, or external clock that drives the node.

JTAG interface

The JTAG interface supports a teststation and a mechanism to fanout JTAG to all the boards in a node. It is used only for testing.

JTAG functions are described in the following sections.

Teststation interface

The teststation can be a PA-RISC based workstation. The interface to the teststation is an ethernet AUI port for flexibility in connecting to many workstations. It is also easily expandable.

DC test of a node

To perform the `DC test,` the Test Bus Controller (TBC) first scans data to all boards in a node. Then each JTAG device performs a capture step that completes the movement of the test data from the driver to the receiver. This step is described in the JTAG 1149.1 specification.

AC test of a node

To perform the `AC test,` the Test Bus Controller (TBC) scans data to all boards in a node and then loads an `AC test` instruction into all ASICs on one board at a time. The scan ring on each board is paused.

Once all boards have been loaded with the `AC test` instruction, the TBC takes all boards out of pause mode simultaneously, causing them all to exit update together and execute the `AC test`.

The `AC test` enables clocks inside the ASICs so that they test internal and external paths at the system clock rate. They all execute on the same system clock.

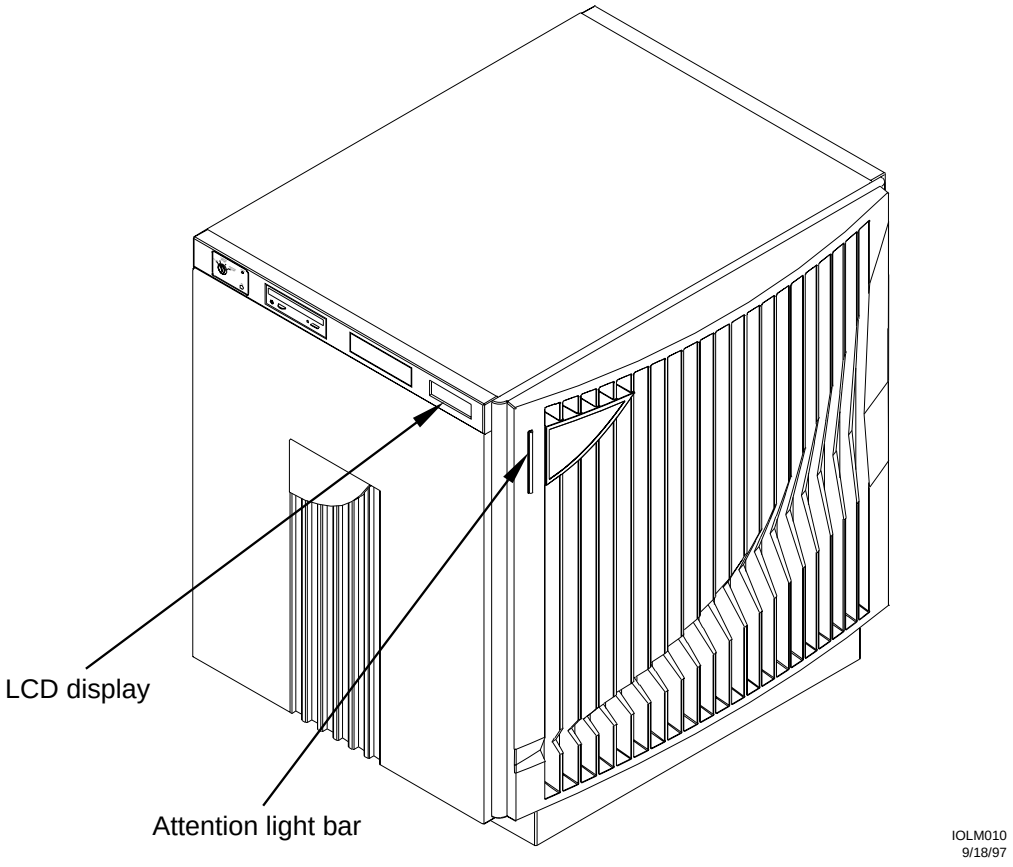
JTAG fanout

The teststation interface is thin ethernet. In addition to the teststation, this port is also used for the console ethernet. There is one cable that connects to all the nodes and to the teststation (if it exists) and to whatever device or network that will display the console.

System displays

The V2500 server provides two means of displaying status and error reporting: an LCD and an Attention light bar.

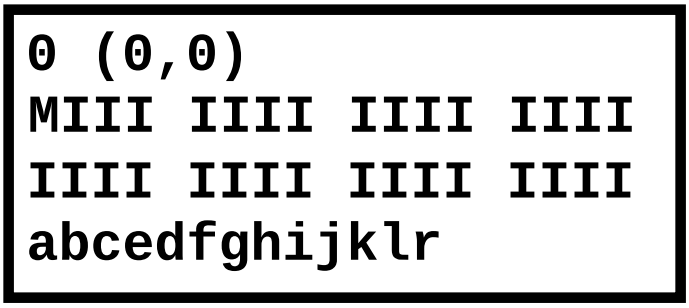
Figure 3 System displays



Front panel LCD

The front panel is a 20-character by 4-line liquid crystal display as shown in Figure 4.

Figure 4 **Front panel LCD**



When the node key switch is turned on, the LCD powers up but is initially blank.

Power-On Self Test (POST) starts displaying output to the LCD. The following illustrates this output shown in Figure 4:

Node status line

The Node Status Line shows the node ID in both decimal and X, Y topology formats.

Processor status line

The processor status line shows the current run state for each processor in the node. Table 2 shows the initialization step code definitions and Table 3 shows the run-time status codes. The M in the first processor status line stands for the monarch processor.

Table 2 **Processor initialization steps**

Step	Description
0	Processor internal diagnostic register initialization
1	Processor early data cache initialization.
2	Processor stack SRAM test.(optional)
3	Processor stack SRAM initialization.
4	Processor BIST-based instruction cache initialization.
5	Processor BIST-based data cache initialization

Step	Description
6	Processor internal register final initialization.
7	Processor basic instruction set testing. (optional)
8	Processor basic instruction cache testing. (optional)
9	Processor basic data cache testing. (optional)
a	Processor basic TLB testing (optional)
b	Processor post-selftest internal register cleanup. (optional)

Table 3 **Processor run-time status codes**

Status	Description
R	RUN: Performing system initialization operations.
I	IDLE: Processor is in an idle loop, awaiting a command.
M	MONARCH: The main POST initialization processor.
H	HPMC: processor has detected a high priority machine check (HPMC).
T	TOC: processor has detected a transfer of control (TOC).
S	SOFT_RESET: processor has detected a soft RESET.
D	DEAD: processor has failed initialization or selftest.
d	DECONFIG: processor has been deconfigured by POST or the user.
-	EMPTY: Empty processor slot.
?	UNKNOWN: processor slot status in unknown.

Message display line

The message display line shows the POST initialization progress. This is updated by the monarch processor. The system console also shows detail for some of these steps. Table 4 shows the code definitions.

Table 4 **Message display line**

Message display code	Description
a	Utilities board (SCUB) hardware initialization.
b	Processor initialization/selftest rendezvous.
c	Utilities board (SCUB) SRAM test. (optional)
d	Utilities board (SCUB) SRAM initialization.
e	Reading Node ID and serial number.
f	Verifying non-volatile RAM (NVRAM) data structures.
g	Probing system hardware (ASICs).
h	Initializing system hardware (ASICs).
i	Probing processors.
j	Initialing, and optionally testing, remaining SCUB SRAM.
k	Probing main memory.
l	Initializing main memory.
r	Enabling system error hardware.

Power supply indicators

When the keyswitch on the operator panel is in the **DC ON** position both the AC power (amber) LED and the DC power (green) LED on each of the power supplies should be on.

Attention light bar

The Attention light bar is located at the top left corner on the front of the HP 9000 V2500 server as shown in Figure 3 on page 12. This light bar displays system status in three ways:

- Off—system powered down
- Steady on—system powered up
- Flashing—error condition

The SMUC prioritizes system environmental errors and warnings and passes the information to the power-on circuit. This circuit prioritizes the 6-bit field with its environmental conditions and produces a 7-bit field plus an attention bit (ATTN) that drives the attention light bar. ATTN is on if there is an environmental warning. Second-level registers in the SMUC drive the attention light bar.

In general, the power-on-detected errors are a higher priority than SMUC-detected errors, the lower the error code number, the higher its priority. Environmental warnings are lower priority than the environmental errors. Table 5 shows the attention light bar error codes in hexadecimal. The top of the table is the highest priority, the bottom the lowest. If a higher condition occurs, that one is displayed.

Table 5 Environmental attention light bar

ATTN bit	attention light bar	Description
1	00	SCUB 3.3-V error (highest priority)
1	01	ASIC Install 0 (MIB)
1	02	ASIC Install 1 (MEM)
1	03	FPGA not OK
1	04-07	DC OK error (UL, UR, LL, LR)
1	08-11	48-V error, NPSUL fail, PWRUP=0-9
1	12-1B	48-V error, NPSUR failure, PWRUP=0-9
1	1C-25	48-V error, NPSLL failure, PWRUP=0-9

ATTN bit	attention light bar	Description
1	26-2F	48-V error, NPSLR failure, PWRUP=0-9
1	30-39	48-V error, no supply failure, PWRUP=0-9
1	3A	48-V yo-yo error
1	3B	MIB power failure (PB)
1	3C	Clock failure
1	3D-3F	Not used (3)
1	40-47	MB0-MB7 power failure
1	48-4F	PB0L, PB1R, PB2L, PB3R, PB4L, PB5R, PB6L, PB7R power failure
1	50-57	PB0R, PB1L, PB2R, PB3L, PB4R, PB5L, PB6R, PB7L power failure (possibly switch R and L)
1	58-5B	IOB (LF,LR,RF,RR) power failure
1	5C-61	Fan failure (UR,UM,UL,LR,LM,LL)
1	62	Ambient hot or ambient shutdown
1	63	Overtemp MIB
1	64-67	Overtemp quadrant (RL, RU, LL, LU)
1	68	Hard error
1	69	Ambient warm
1	6A-6F	Not used (6)
1	70-73	DC supply maintenance (UL,UR,LL,LR)
1	74	AC circuit failure
1	75-7F	Not used (11)
0	00-09	PWRUP state (00=System all powered up), attention LED off

SCUB 3.3-Volt error

This error indicates that the SCUB 3.3-Volt power supply has failed, but the 5-Volt supply has not.

ASIC installation error

Each ASIC in the node has ASIC Install lines to prevent power-up if an ASIC is installed incorrectly (such as a SPAC installed in an ERAC position). If an ASIC is improperly installed, the Utilities board does not power up the system. This condition is not monitored after power up.

DC OK error

When this error is displayed, the power-on circuit did not power up the system, because one or more 48-Volt power supplies reported an error. In systems with redundant 48-Volt power supplies, this error means that two or more 48-Volt supplies reported an error.

48-Volt error

If the 48-Volt supply has dropped below 42 volts for any reason other than normally turning off the system or an ac failure, then this error is displayed by the power-on circuit. Also, the 48-Volt supply that reported the error and the power-up state of the system at the time of the error is displayed.

48-Volt yo-yo error

This error indicates that a 48-Volt error occurred and the SCUB lost and then later regained power without the machine being turned off. The power-on circuit will display this error and not power on the system, because the 48-Volt supply is likely at fault.

Clock failure

If the system clock fails, the SMUC is unable to monitor environmental errors that could possibly damage the system. If the power-on circuit receives no response from the SMUC, it powers down the system and displays this error.

FPGA configuration and status

The SMUC is programmed by a serial data transfer from EEPROM upon utility board power-up. If the transfer does not complete properly, the SMUC cannot configure itself and many environmental conditions cannot be monitored. The power-on circuit monitors both the SMUC and SPUC and does not power up the system, if they are not configured correctly.

Board over-temperature

On each board in the node, there is one temperature sensor that detects board overheating. The sensors are bussed together into four-node quadrants, along with the MIB, and applied to the SMUC.

Fan sensing

The V2500 node has up to six fans, but only four may be configured. Sensors in the fans determine if the fans are running properly. The SMUC waits 12.8 seconds for the fans to spin up after power-up before monitoring them. It is assumed that the unconfigured fans do not report errors.

Power failure

Because a power failure on a board could cause damage to other boards, a mechanism on each board detects 3.3-Volt failures on each board. Power failures are considered environmental errors, and the system is powered down after they are detected.

MidPlane Interface Board (MIB) power failure

If the MIB power fails, the power-on circuit powers down the entire node. The Utilities board is still active, but the power-on circuit displays the power failure condition and disables all Utilities board outputs that drive the node. This condition persists until power is cycled on the Utilities board.

48-Volt maintenance

There are four 48-Volt power supplies; three are required, and one is a redundant source. Each sends a signal to the power-on circuit. If any supply fails at any time, the circuit asserts the 48-V maintenance line to

the SMUC, which reports the environmental warning to the processors. The power-on circuit displays the “highest priority” 48-Volt supply that failed.

Ambient air sensors

The ambient air sensors detect a too warm or too hot condition in the input air stream to the Utilities board (and therefore the entire node). Ambient air too warm is an environmental warning; ambient air too hot is an environmental error that powers down the system.

The temperature set points are set using the sppdsh utility, described in “sppdsh” on page 268. The digital temperature sensor has nonvolatile storage for the temperature set points. Power-on reset starts the digital temperature sensor without the core logic microprocessor intervening.

AC circuit fail

An AC circuit failure denotes that the circuit that detects AC failures is broken. A power-on reset clears this warning.

2 Configuration management

The teststation allows the user to configure the node using the `ts_config` utility. `ts_config` configures the teststation to communicate with the node. The teststation daemon, `ccmd`, monitors the node and reports back configuration information, error information and general status. `ts_config` must be run before using `ccmd`.

Two additional utilities, `sppdsh` and `xconfig`, allow reading or writing configuration information and changing it. OBP can also be used to modify the configuration.

Teststation

The teststation is used for configuring, monitoring, testing, and error logging. It is not required for normal operation of a node.

The teststation communicates with the JTAG interface in the nodes. The JTAG port remains idle if no teststation is connected to it. It receives communications packets, interprets requests, and generates responses to them. The hardware on the node can read board information, system configuration, device revisions, and environmental conditions. When a teststation is present, all of these parameters are read or written by the configuration management tools.

The configuration management daemon, `ccmd`, initiates communications between the teststation and the nodes.

ts_config

```
ts_config [-display display name]
```

V2500 nodes added to the teststation must be configured by `ts_config` to enable diagnostic and scan capabilities, environmental and hard-error monitoring, and console access.

Once the configuration for each node is set, it is retained when new teststation software is installed.

`ts_config` tasks include:

- Configuring a node—Adding and removing a node to the teststation configuration
- Configuring the terminal mux—Configuring and removing the terminal mux on the teststation
- Installing a node—Upgrading JTAG firmware, configuring a node scub_ip address, and resetting a node

The user must have root privilege to configure a node of the terminal mux, because several HP-UX system files are modified during the configuration.

Starting ts_config

To start `ts_config` from the teststation desktop, click on an empty area of the background to obtain the Workspace menu and then select the `ts_config (root)` option. Enter the root password.

To start `ts_config` from a shell (local or remote), ensure that the DISPLAY environment variable is set appropriately before starting `ts_config`.

For example:

```
$ DISPLAY=myws:0; export DISPLAY     (sh/ksh/sppdsh)
% setenv DISPLAY myws:0       (csh/tcsh)
```

Also, the `-display` start-up option may be used as shown below:

For example:

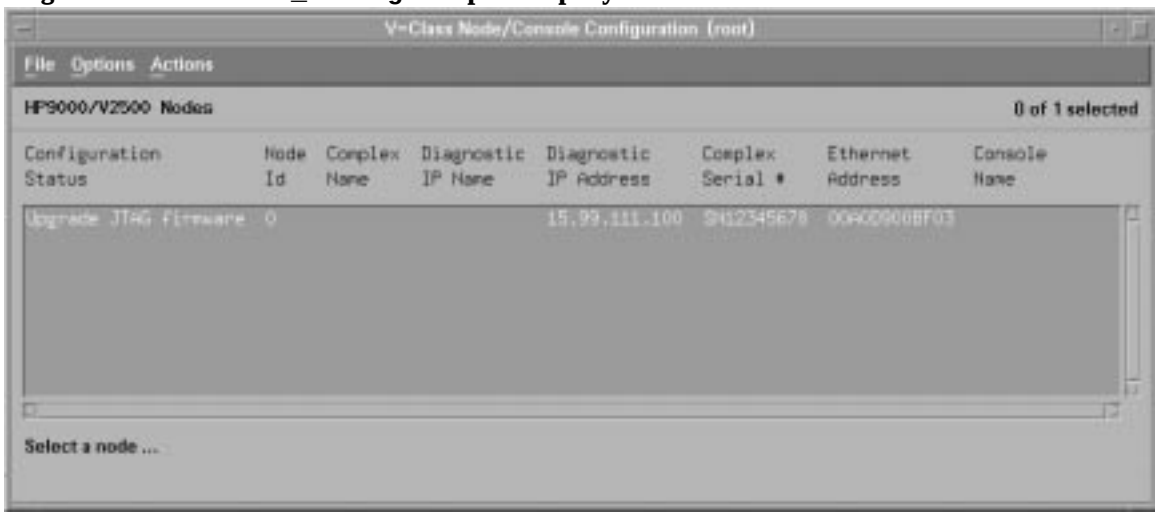
```
# /spp/bin/ts_config  -display myws:0
```

NOTE For shells that are run from the teststation desktop, the DISPLAY variable is set (at the shell start-up) to the local teststation display.

ts_config operation

The `ts_config` utility displays an active list of nodes that are powered up and connected to the teststation diagnostic LAN. The operator selects a node and configures the selected node. A sample display is shown below.

Figure 5 `ts_config` sample display



The window has two main parts: the drop-down menu bar and the display panel. The display panel contains a list of nodes and their status. To select a node, click with the left-mouse button the line containing the desired node entry in the list. When a node is selected, information about that node is shown at the bottom of the `ts_config` window. If an action needs to be performed to configure the node, specific instructions are included.

`ts_config` automatically updates the display when it detects either a change in the configuration status of any node or a new node. However, the automatic update is disabled while the user has a node selected.

After the node is selected, the display is not updated until the user selects an action or refreshes the node list. The upper right corner of the `ts_config` window indicates whether a node has been selected.

The ts_config window title includes in parenthesis the name of the effective user ID running ts_config, either root or sppuser.

The ts_config display shows the configuration status of the nodes. Table 6 shows the possible status values.

Table 6 ts_config status values

Configuration Status	Description	Action Required
Upgrade JTAG firmware	The version of JTAG firmware running on the SCUB does not support the capabilities required to complete the node configuration process.	Select the node and follow the instructions given at the bottom of the ts_config window. ts_config guides the operator through the JTAG firmware upgrade procedure.
Not Configured	ts_config has detected the node on the Diagnostic LAN and the JTAG firmware is capable of supporting the node configuration activity and the node needs to be configured.	Select the node and follow the instructions given at the bottom of the ts_config window. ts_config guide the operator through the node configuration procedure described later in this document.

Configuration Status	Description	Action Required
Active	The node is configured and answering requests on the Diagnostic LAN.	None required. This is the desired status.
Inactive	The teststation node configuration file contains information about the specified node, but the node is not responding to requests on the Diagnostic LAN.This status is also shown if a node was configured and then removed from the teststation LAN without being deconfigured.	Power-up the node and/or check for a LAN connection problem. If the node information shown is for a node that has been removed, select the node then select “Actions,” “Deconfigure Node,” and click “Yes.”
Node Id changed	The node is configured and answering requests on the diagnostic LAN, but the node ID currently reported by the node does not match the teststation configuration information.	Select the node to obtain additional information. If the node COP information was changed to a different node ID and the new node ID is correct, select “Actions,” “Configure Node,” then click “Configure.” The teststation configuration information is updated using the new node ID.

Configuration Procedures

NOTE

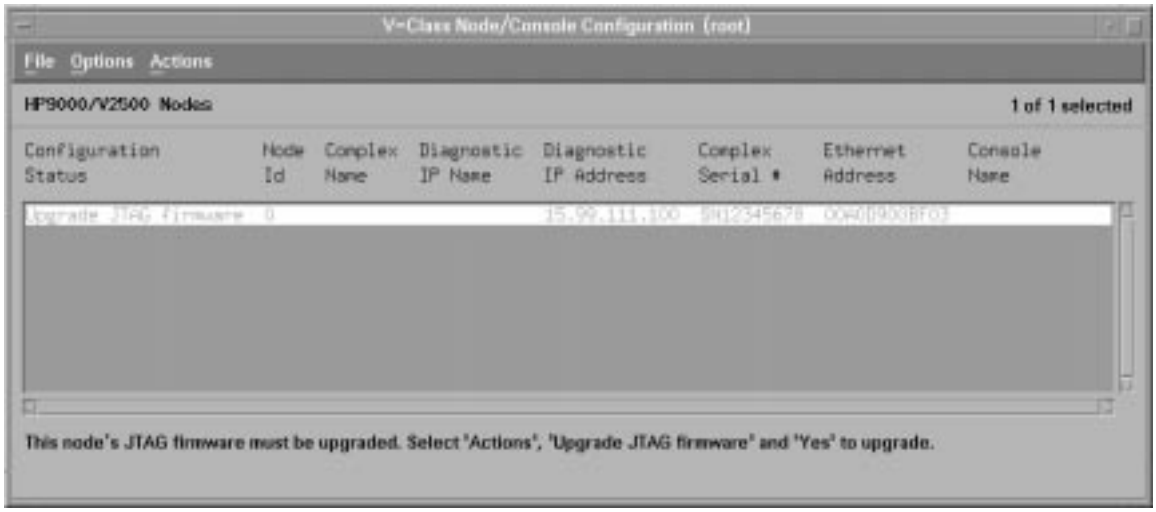
This procedure does not need to be performed unless the status shows “Upgrade JTAG firmware.” If the node shows “Not Configured,” skip this section.

The following procedures provide additional details about each configuration action. `ts_config` automatically guides the user through the appropriate procedure when a node is selected.

Upgrade JTAG firmware

Step 1. Select the node from the list in the display panel. For example, clicking on node 0 in the list highlights that line as shown in Figure 6.

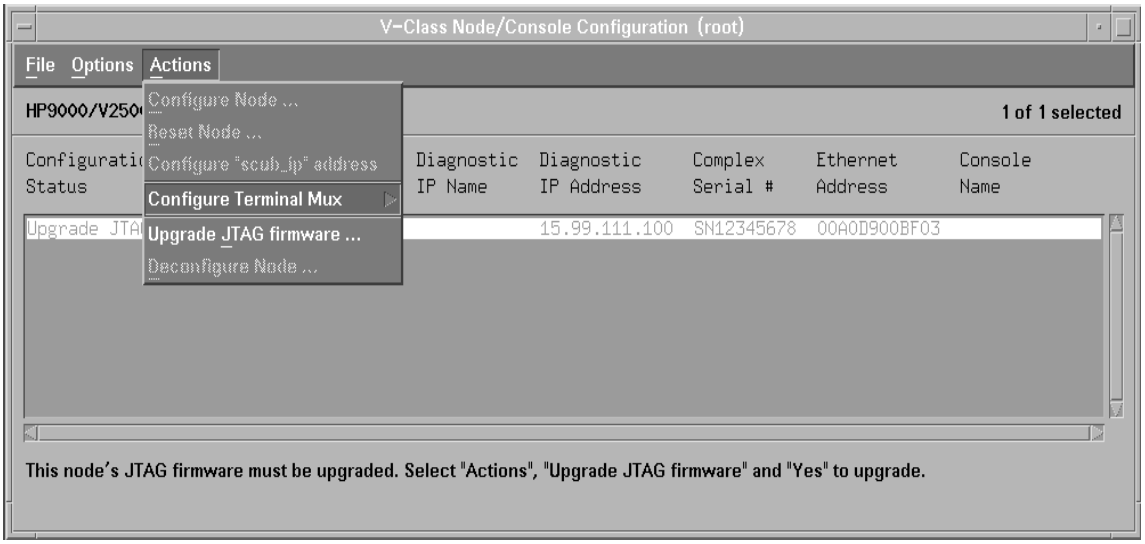
Figure 6 ts_config show node 0 highlighted



Notice that after the node has been highlighted that `ts_config` displays information concerning the node. In this step, it tells the user what action to take next, “This node's JTAG firmware must be upgraded. Select “Actions,” “Upgrade JTAG firmware” and “Yes” to upgrade.”

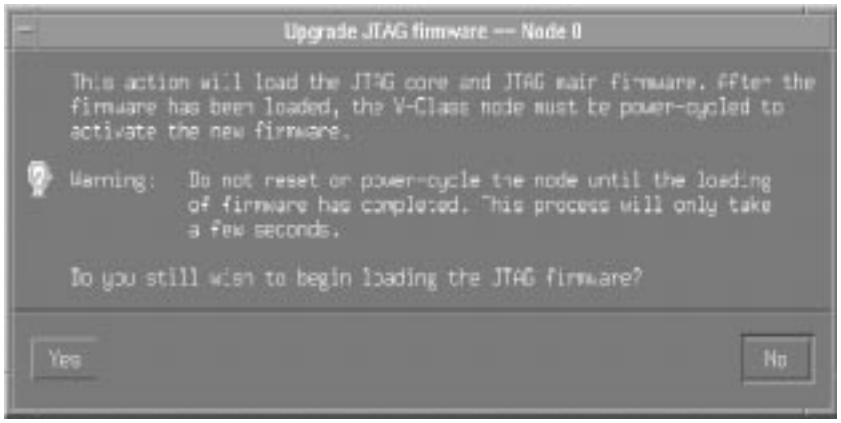
Step 2. Select “Actions” to drop the pop-down menu and then click “Upgrade JTAG firmware,” as shown in Figure 7.

Figure 7 **ts_config “Upgrade JTAG firmware” selection.**



Step 3. A message panel appears as the one shown in Figure 8. Read the message. If this is the desired action, click “Yes” to begin the upgrade.

Figure 8 **Upgrade JTAG firmware confirmation panel**



Step 4. After the firmware is loaded a panel appears as the one shown in Figure 9. Click “OK” and then power-cycle the node to activate the new firmware.

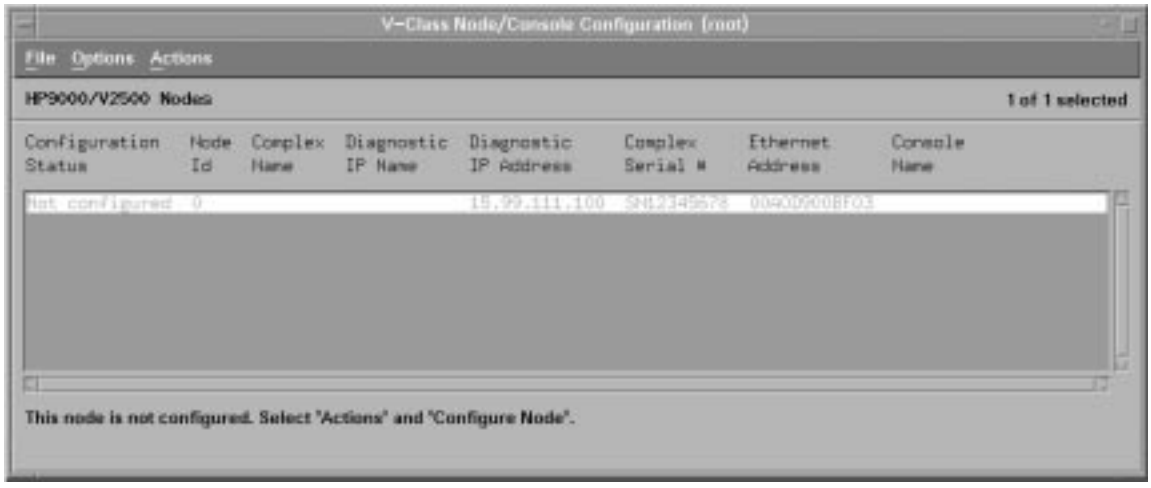
Figure 9 **ts_config power-cycle panel**

When the node is powered up, the “Configuration Status” should change to “Not Configured.”

Configure a Node

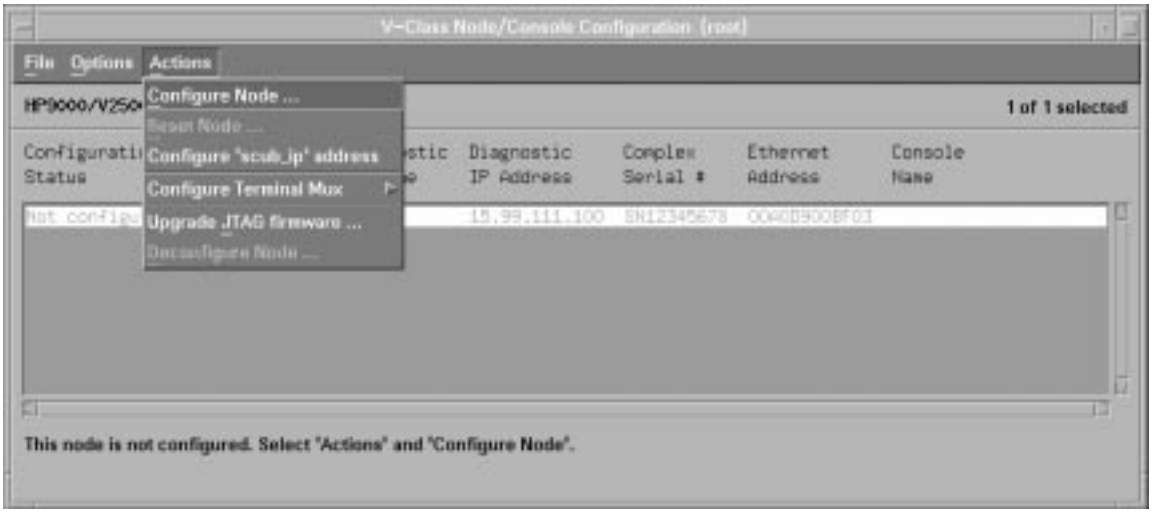
Step 1. Select the desired node from the list of available nodes. When the node is selected, the appropriate line is highlighted as shown in Figure 10. Notice the bottom of the display indicates the Node 0 is not configured and provides the steps necessary to configure the node.

Figure 10 **ts_config indicating Node 0 as not configured**



Step 2. Select “Actions” and then click “Configure Node,” as shown in Figure 11.

Figure 11 **ts_config “Configure Node” selection.**



After invoking `ts_config` to configure the node, a node configuration panel appears as the one in Figure 12.

Figure 12 **ts_config node configuration panel**

Step 3. Enter a name for the V2500 System. The teststation uses this name as the “Complex Name” and to generate the IP hostnames of the Diagnostic and OBP LAN interfaces. Select a short name that teststation users can easily relate to the associated system (for example: **`hw2a`**, **`swtest`**, etc.).

Step 4. Select an appropriate serial connection for the V2500 console from the pop-down option menu in the node configuration panel.

`ts_config` automatically assigns the first unused serial port. If the terminal mux has been configured, the terminal mux ports are included in the list of available serial connections.

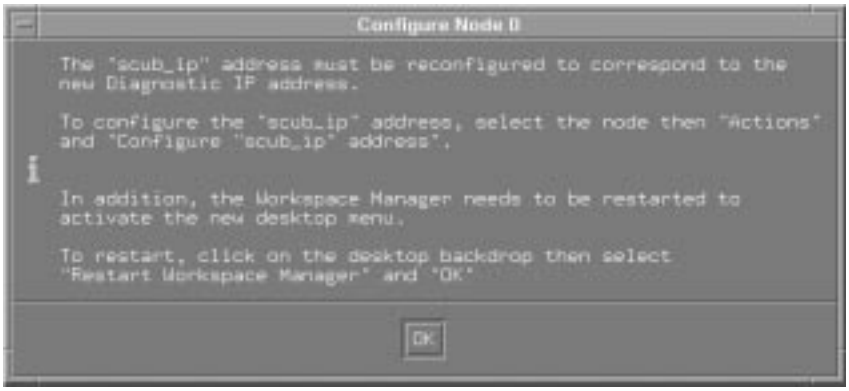
The IP address information for the Diagnostic interface is provided. The `ts_config` utility automatically changes the IP address of the diagnostic LAN interface to prevent a duplicate when other nodes are added to this teststation configuration.

`ts_config` automatically updates the local /etc/hosts file with the names and addresses of the Diagnostic and OBP LAN interfaces.

Step 5. Click “Configure.”

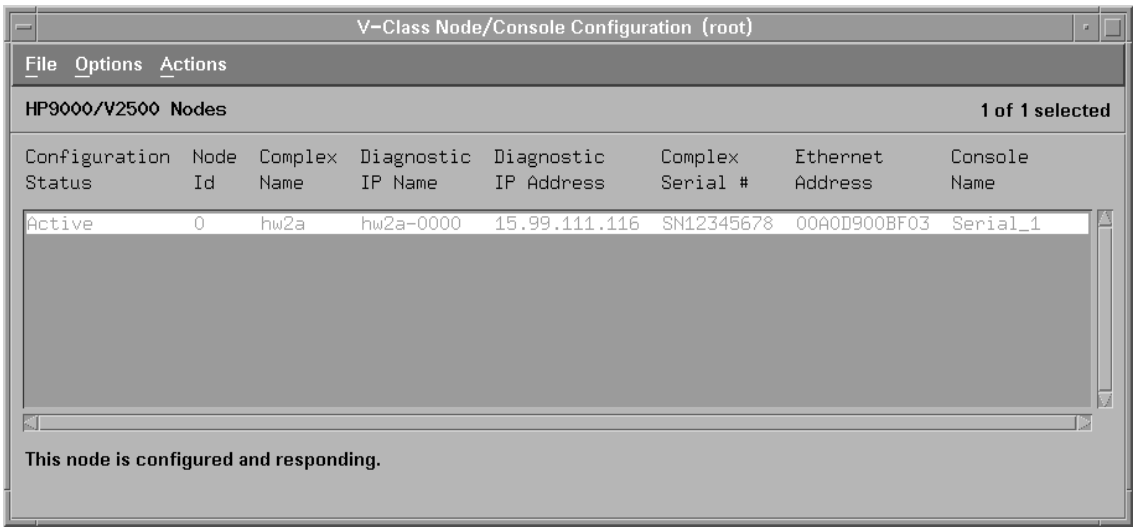
This updates several teststation files. The node configuration confirmation panel appears as the one in Figure 13.

Figure 13 `ts_config` node configuration confirmation panel.



Step 6. Read the panel and click “OK.” When the configuration process is complete, the “Configuration Status” of the node changes to “Active,” as shown in Figure 14.

Figure 14 `ts_config` indicating Node 0 is configured



Step 7. Restart the Workspace Manager: Click the right-mouse button on the desktop background to activate the root menu. Select the “Restart” or “Restart Workspace Manager” option, then “OK” to activate the new desktop menu.

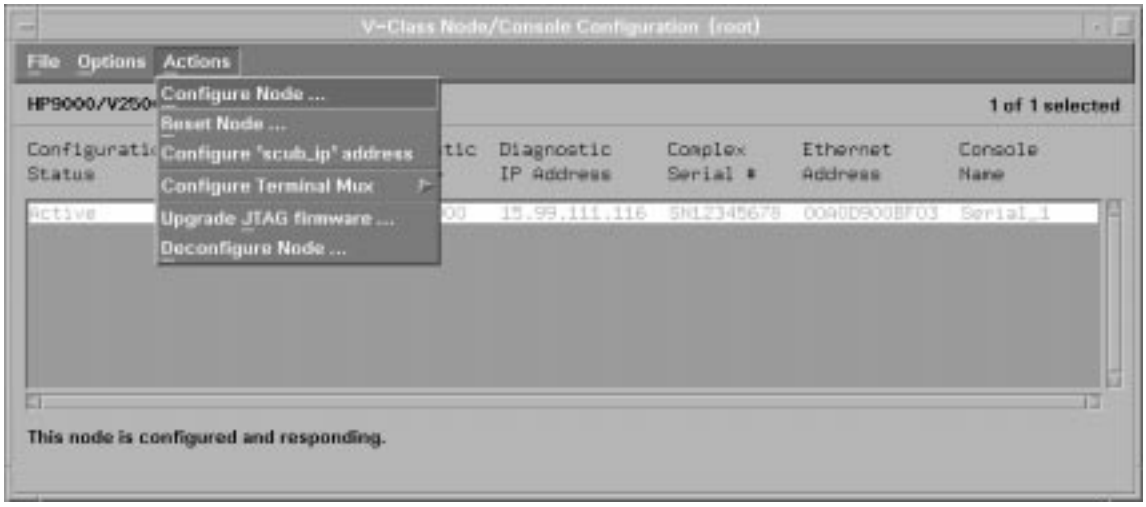
NOTE If adding multiple nodes to the teststation, wait until the final node is added before restarting the Workspace Manager.

Configure the “scub_ip” address

Step 1. Select the desired node from the list of available nodes.

Step 2. In the `ts_config` display panel, select “Actions” and then “Configure ‘scub_ip’ address,” as shown in Figure 15.

Figure 15 ts_config “Configure ‘scub_ip’ address” selection

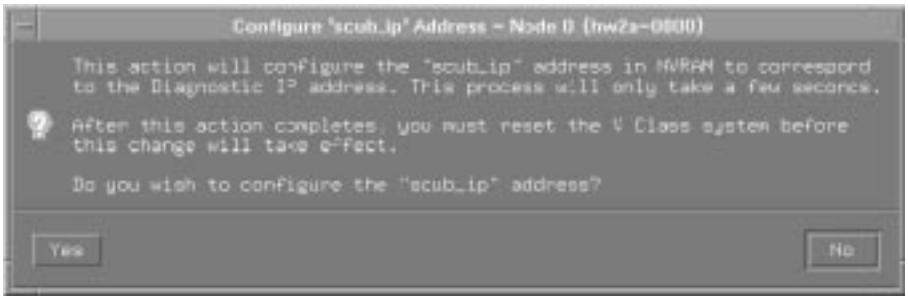


`ts_config` checks the scub_ip address stored in NVRAM in the node.

If the scub_ip address is correct, no action is required. If the node is not detected and scanned by `ccmd`, `ts_config` may ask you to try again later. The `ccmd` detection scan process should take less than a minute.

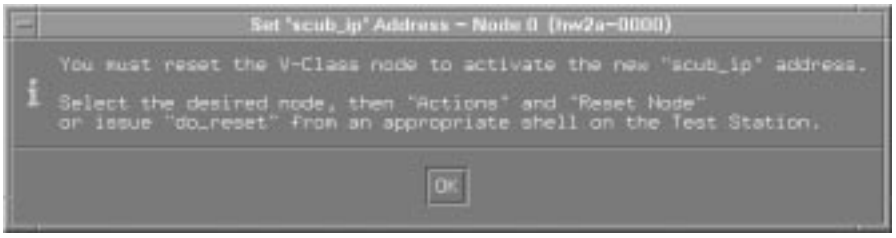
Step 3. If prompted by `ts_config` (as indicated by the panel in Figure 16), click “Yes” to correctly set the scub_ip address.

Figure 16 ts_config scub_ip address configuration confirmation



Step 4. A panel as the shown in Figure 17 appears confirming that the scub_ip address is set. Click OK.

Figure 17 **ts_config scub_ip address set confirmation panel**



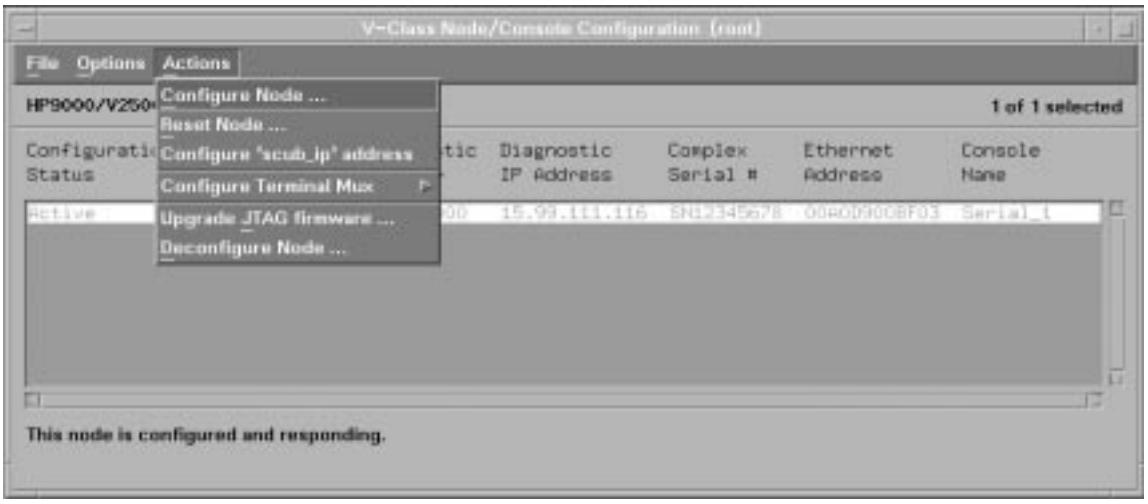
Initiate a node reset to activate the new scub_ip address.

Reset the Node

Step 1. Select the desired node from the list of available nodes.

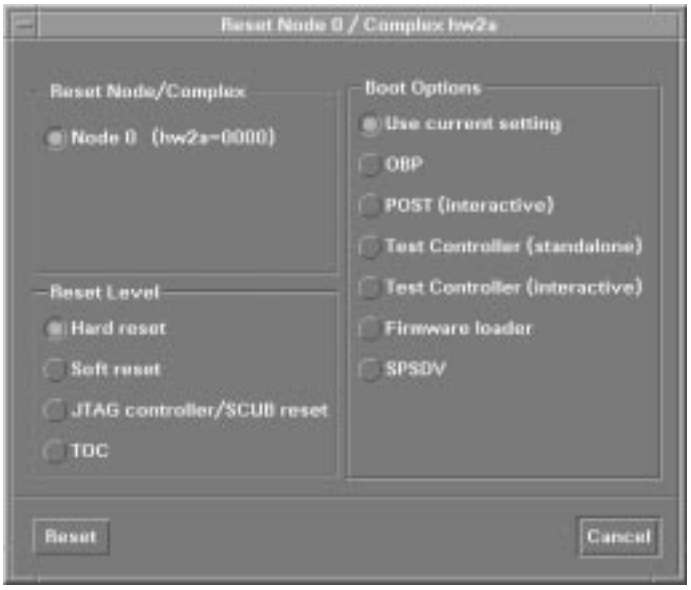
Step 2. Select “Actions,” then “Reset Node.” This is indicated in Figure 18.

Figure 18 **ts_config “Reset Node” selection**



A panel as the one shown in Figure 19 appears.

Figure 19 **ts_config node reset panel**



Step 3. In the Node Reset panel, select the desired “Reset Level” and “Boot Options,” then click Reset.”

Deconfigure a Node

Deconfiguring a node removes the selected node from the teststation configuration. The teststation will no longer monitor the environmental and hard-error status of this node. Console access to the node is also be disabled.

Step 1. Select the desired node from the list of available nodes.

Step 2. Select “Actions,” then “Deconfigure Node,” then click “Yes.”

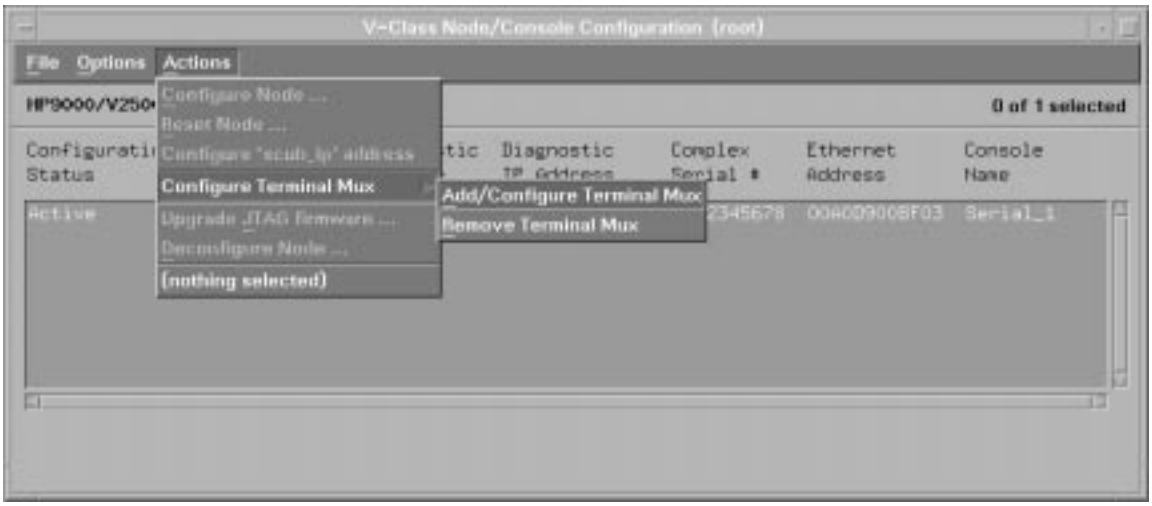
Add/Configure the Terminal Mux

To add or reconfigure the terminal mux, perform the following procedure.

Step 1. In the `ts_config` display, select “Actions,” then “Configure Terminal Mux.”

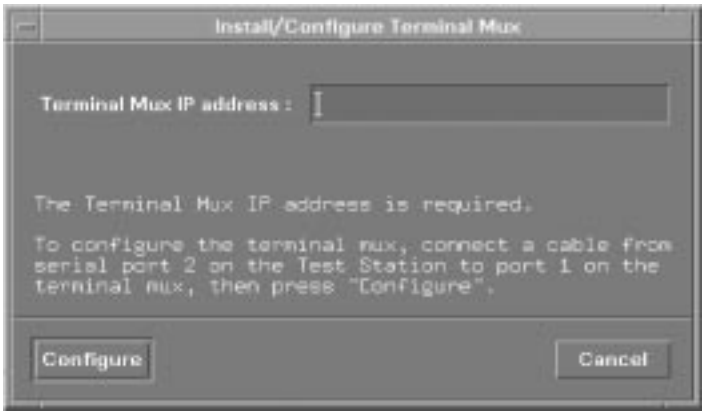
Step 2. Select “Add/Configure Terminal Mux.” This is indicated in Figure 20.

Figure 20 **ts_config “Add/Configure Terminal Mux” selection.**



A panel appears as the on shown Figure 21. This panel requires the terminal mux IP address.

Figure 21 **ts_config terminal mux IP address panel**



Step 3. Connect a serial cable from serial port 2 on the teststation to port 1 on the terminal mux.

Step 4. Enter the desired “Terminal Mux IP Address” and click “Configure,” as indicated in Figure 22.

Figure 22 **Terminal mux IP address entered into panel**

Remove terminal mux

`ts_config` does not remove the terminal mux if any node consoles are assigned to terminal mux ports.

Step 1. Select “Actions,” then “Configure Terminal Mux.”

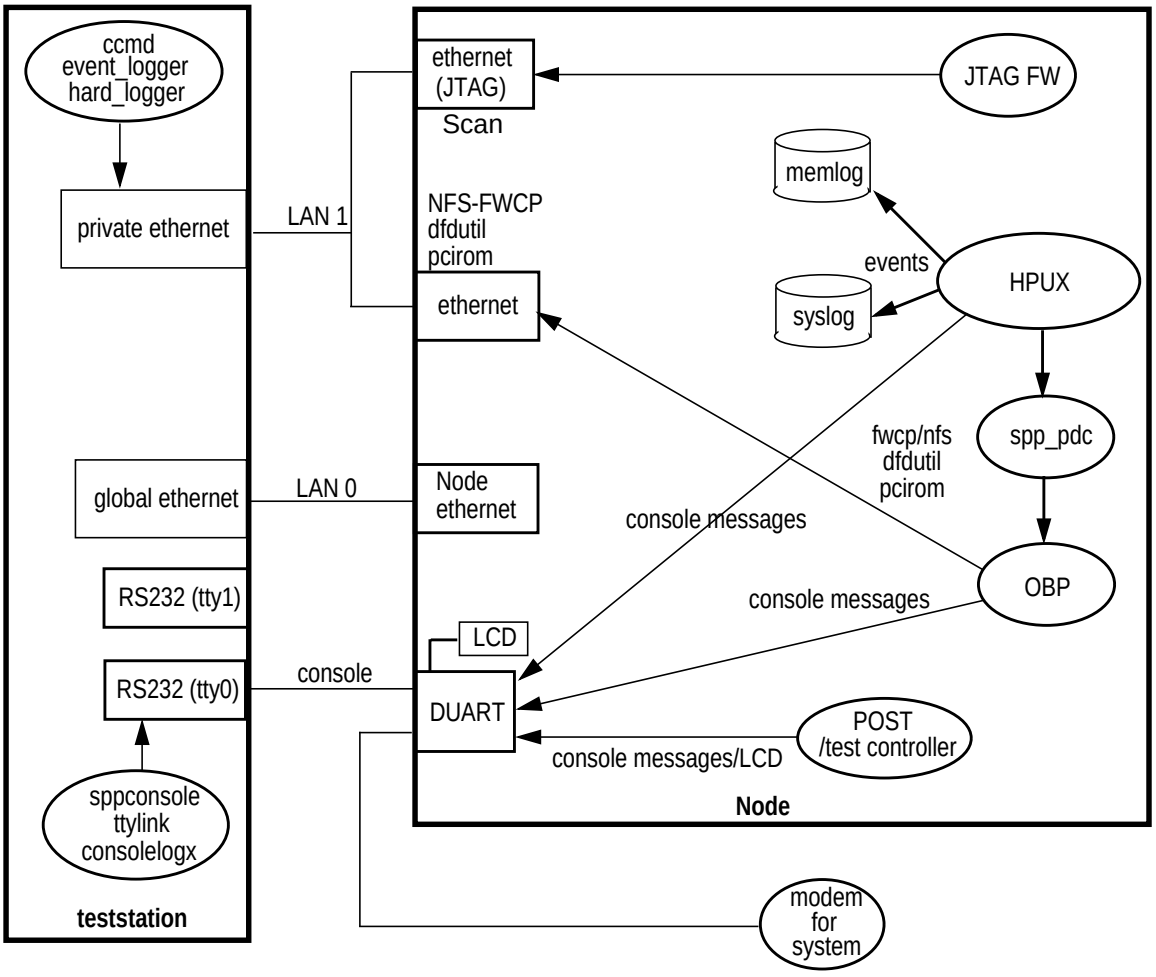
Step 2. Select “Remove Terminal Mux,” then click “Yes.”

Teststation-to-system communications

This section describes how the teststation communicates with the system using the utilities presented in Chapter 11, “Utilities.”

Figure 23 depicts the V-Class server to teststation communications using HP-UX.

Figure 23 Teststation-to-system communications



The hardware components located on the SCUB are shown in the diagram on the left side of the node or system. They include three ethernet ports and one DUART.

A layer of firmware between HP-UX and OBP called `spp_pdc` allows the HP-UX kernel to communicate with OBP. `spp_pdc` is platform-dependent code and runs on top of OBP providing access to the devices and OBP configuration properties.

LAN communications

One system ethernet port connects to global LAN 0. The other ethernet port connects to the system private LAN 1. The JTAG port is used for scanning. The other port is used for downloading system firmware via `nfs` using `fwcp`, via `tftp` using `pdcfl`, downloading disk firmware using `dfdutil` (`dfdutil` uses `tftp` for read peripheral firmware), and loading Symbios FORTH code using `pcirom`.

The configuration deamon, `ccmd`, which is located on the teststation obtains system configuration information.

Serial communications

The DUART port on the SCUB provides an RS232 serial link (tty 0) to the teststation. Through this port HP-UX, OBP, POST and the Test Controller send console messages. The teststation processes these message using `sppconsole`, `ttylink`, and `consolelogx`. POST and OBP also send system status to the LCD connected to the DUART.

ccmd

ccmd builds a configuration information database on the teststation. The board names and revisions, the device names and revisions, and the start-up information generated by POST are all read and stored in memory for use by other diagnostic tools.

ccmd is typically run automatically from /etc/inittab on the teststation.

Entering **init** on the teststation starts ccmd. init monitors ccmd and respawns it if it ever stops. Once started, ccmd becomes a daemon and allocates a block of teststation memory used for a database for all nodes, boards, and devices.

ccmd broadcasts a command to all JTAG ports to report in. Each node responds with its IP address, error status, complex identification, and the node identification number. ccmd continues until no new responses are detected.

Once ccmd has all valid node numbers and IP addresses, it scans each ring of each node looking for JTAG device IDs. The JTAG IDs contain device and revision information stored in the teststation database. The JTAG ID is cross checked in the /spp/data/part_ids file to retrieve a complete device description of the part. The file for the part is also in /spp/data.

After ccmd loads all parts and their descriptions into the database, it reads all board information. Board and device information is cross checked in the file /spp/data/DB_RING_FILE. Complex and node configuration files for est_config is written to the /spp/data/ directory.

Once all information is in place, ccmd monitors the node for changes in node configuration or error status. After a 10 second pause, ccmd once again broadcasts to determine which nodes are powered up or have an error condition. If a node response is not received after six broadcasts, then the node is removed from the database of existing nodes.

If no nodes are responding, ccmd clears all node data and continues broadcasting, waiting for a node to respond. If a node powers up, the nodes database is rebuilt.

If ccmd detects a hard error, it starts the hard_logger script to extract additional information from the node through the JTAG interface. After the hard_logger runs, ccmd resets the node or complex that failed. This behavior can be stopped with autoreset.

ccmd sends output to the console. If running under X-windows as sppuser, it sends its output to the teststation console message output window. The -d debug option generates a substantial amount of console output. Output is also sent to the file /spp/data/ccmd_log.

ccmd does respond to several signals. The SIGHUP signal tells ccmd to rebuild the teststation database. A sigint or sigabort signal terminates the ccmd process.

NOTE The time zone information is read when ccmd starts. If the time zone information changes, ccmd should be restarted as well.

xconfig

`xconfig` is the graphical tool that can also modify the parameters initialized by POST to reconfigure a node.

The graphical interface allows the user to see the configuration state. Also the names are consistent with the hardware names, since individual configuration parameters are hidden to the user. The drawback of `xconfig` is that it can not be used as a part of script-based tests, nor can it be used for remote debug.

`xconfig` is started from a shell. Information on node 0 is read and interpreted to form the starting X-windows display shown in Figure 24.

The xconfig window appears on the system indicated by the environmental variable $DISPLAY. This may be overridden, however, by using the following command:

```
% xconfig -display system_name:0.0
```

The xconfig window has two display views: one shows each component as a physical location in the server, the other shows them as logical names. Figure 24 and Figure 25 show the window in each view, respectively. To switch between view, click on the Help button in menu bar and then click the Change names option. See “Menu bar” on page 45.

Figure 24 xconfig window—physical location names

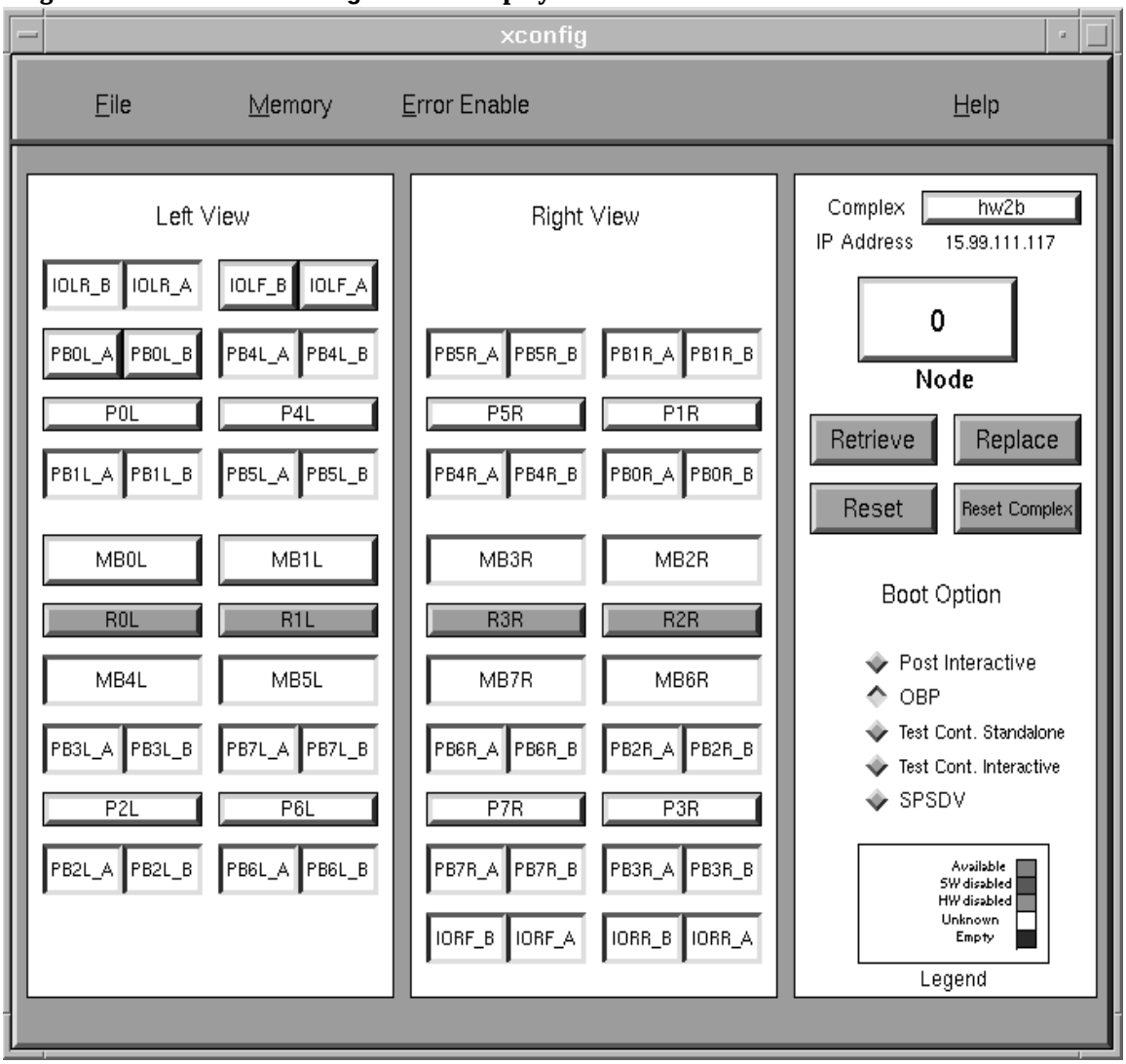
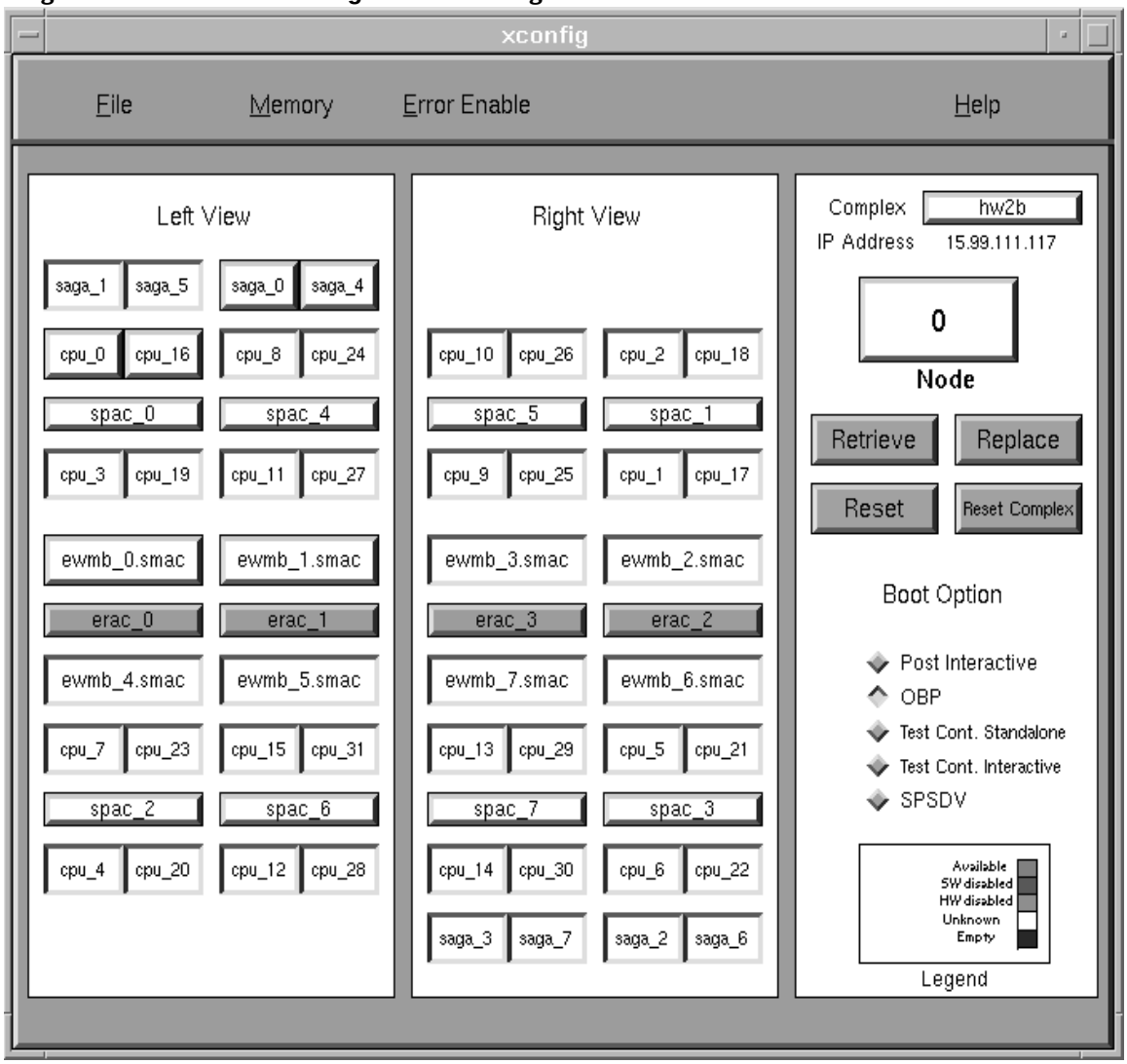


Figure 25 xconfig window—logical names



As buttons are clicked, the item selected changes state and color. There is a legend on the screen to explain the color and status. The change is recorded in the teststation's image of the node.

When the user is satisfied with the new configuration, it should be copied back into the node, and the node should be reset to enable the changes.

The main xconfig window has three sections:

- Menu bar—Provides additional capability and functions.
- Node configuration map—Provides the status of the node.
- Node control panel—Provides the capability to select a node and control the way data flows to it.

Menu bar

The menu bar appears at the top of the xconfig main window. It has four menus that provide additional features:

- File menu—Displays the file and exit options.
- Memory menu—Displays the main memory and CTI cache memory options.
- Error Enable menu—Displays the device menu options for error enabling and configuration.
- Help menu—Displays the help and about options.

The menu bar is shown in Figure 26.

Figure 26 xconfig window menu bar

The File menu provides the capability to save and restore node images and to exit xconfig.

The Memory menu provides the capability to enable or disable memory at the memory DIMM level by the total memory size and to change the network cache size on a multinode complex.

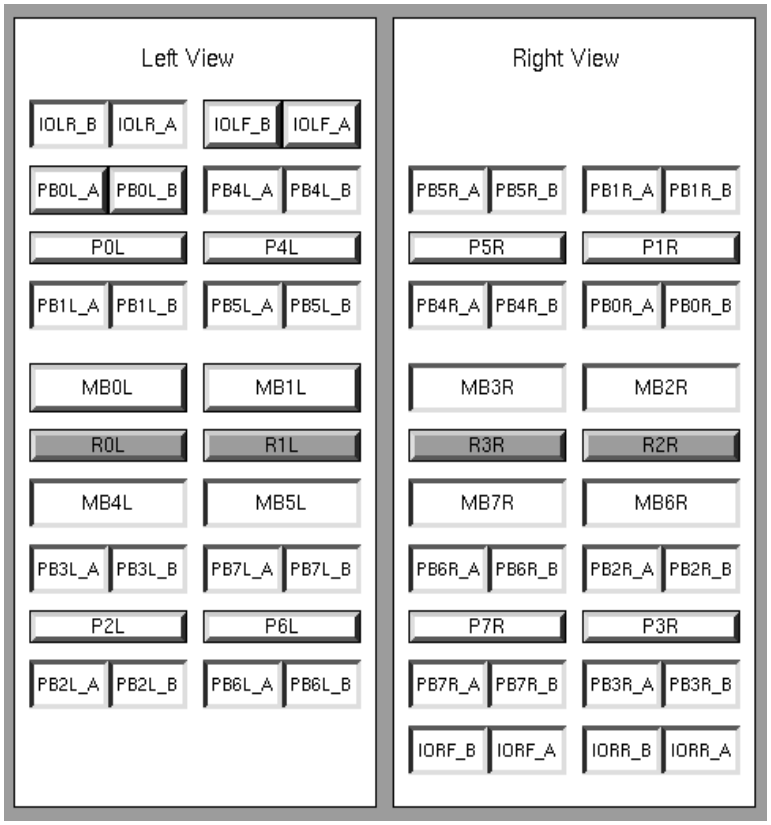
The Error Enable menu provides the capability to change a device's response to an error condition. This is normally only used for troubleshooting.

The Help menu provides a help box that acts as online documentation and also contains program revision information.

Node configuration map

The node configuration map is a representation of the left and right side views of a node as shown in Figure 27.

Figure 27 **xconfig window node configuration map**



The button boxes are positioned to represent the actual boards as viewed from the left and right sides. Each of the configurable components of the node is in the display. The buttons are used as follows:

- Green button—Indicates that the component is present and enabled.
- Red button—Indicates that the component is software disabled in the system.
- White button—Indicates that it is not possible to determine what the status of the component would be if POST were to be started.
- Blue box—Indicates that the component is either not present or fails the power-on self tests.
- Brown button—Indicates that POST had to hardware deconfigure this component in order to properly execute.
- Grey button—Indicates a hardware component that did not properly initialize.

The colors are shown in the legend box of the node control panel.

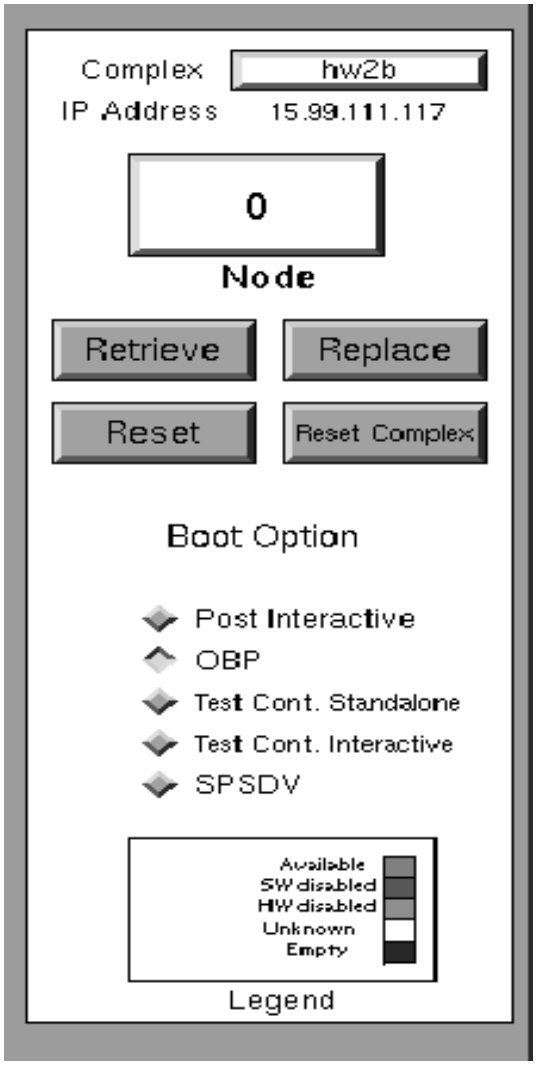
Components can change from enabled to disabled or disabled to unknown by clicking on the appropriate button with the left mouse button.

A multinode system requires an additional component on a memory board to enable the scalable coherent memory interface. This component can be viewed by right clicking the on the memory board button. The right mouse button toggles the memory board display between the memory board and the SCI device

Node control panel

The node control panel allows the user to select a node, select the stop clocks on an error function, select the boot parameters for a node and direct data flow between the node and the `xconfig` utility. It is shown in Figure 28.

Figure 28 **xconfig window node control panel**



The node number is shown in the node box. A new number can be selected by clicking on the node box and selecting the node from the pull-down menu. A new complex can be selected by clicking on the complex box and selecting it from the pull-down. A node IP address is displayed along with the node number and complex.

When a new node is selected and available, its data is automatically read and the node configuration map updated. The data image is kept on the teststation until it is rebuilt on the node using the Replace button. This is similar to the replace command on `sppdsh`.

Even though data can be rebuilt on a node, it does not become active until POST runs again and reconfigures the system. The Reset or Reset All buttons can be used to restart POST on one or all nodes of a system. A multinode system requires a reset all to properly function.

A Retrieve button is available on the node control panel to get a fresh copy of the parameters settings in the system. Clicking this button overwrites the setting local to the teststation and `xconfig`.

The Stop-on-hard button is typically used to assist in fault isolation. It stops all system clocks shortly after an error occurs. Only scan-based operations are available once system clocks have stopped.

The last group of buttons controls what happens after POST completes. The node can become idle or boot OBP, the test controller, or spsdv. The test controller and spsdv are additional diagnostic modes.

Configuration utilities

V2500 diagnostics provides utilities that assist the user with configuration management.

autoreset

`autoreset` allows the user to specify whether ccmd should automatically reset a complex after a hard error and after the hard logger error analysis software has run. `autoreset` occurs if a .ccmd_reset file does not exist in the complex-specific directories

Arguments to `autoreset` arguments include `<complex_name> on` and `<complex_name> off or chk`.

The output of the `chk` option for a complex name of hw2a looks like:

```
Autoreset for hw2a is enabled.
```

or

```
Autoreset for hw2a is disabled.
```

NOTE `autoreset` determines the behavior of `ccmd` when it encounters an error condition. `ccmd` makes its decision whether to reset a complex immediately after running `hard_logger`. Enabling `autoreset` after `hard_logger` has run does not reset the complex.

est_config

`est_config` is a utility that builds the node and complex descriptions used by `est`. `est_config` reads support files at /spp/data/DB_RING_FILE, reads the electronic board identifier (COP chip) and scans to completely describe the node or complex. It also uses the hardware database created by `ccmd`. The data retrieved is organized and sorted into an appropriate node configuration file in the /spp/data/<complex name> directory.

An optional configuration directory can be specified using the `-p` argument. `est_config` works across all nodes unless a specific node or complex is requested with the `-n` option.

NOTE If there is a node_#.pwr file that is older than the node_#.cfg file, existing node configuration files do not need to be updated. `est_config` also generates a complex_uts.cfg file that can be compared against a complex.cfg file for accuracy and consistency.

xsecure

`xsecure` is an application that helps make a V2500 class teststation secure from external sources. This tool disables modem and LAN activity to provide an extra layer of security for the V2500 system. `xsecure` may be run as a command line tool or an windows-based application.

In secure mode, all network LANs other than the tsdart bus are disabled and the optional modem on the second serial port will be disabled. When in normal mode all networks and modems are re-enabled.

If the command line `[-on | -off | -check]` options are used, `xsecure` does not use the GUI interface. These options allow the user to turn the secure mode on, off or allow the user to check the secure mode status

A simple button with a red or green secure mode indicator provides the user with secure mode status information. The red indicator shows that the secure mode process has begun. The label near the red button will inform the user when the test station is secure. A green indicator and the appropriate label shows that the network is available and the teststation may be accessed through the ethernet port.

In order for `xsecure` to work properly the teststation, console cables, terminal mux and modems must be configured in specific ways. The teststation JTAG connections, OBP connections and an optional terminal mux must all be connected to the Diagnostic LAN and identified in the /etc/hosts file as tsdart-d. The sppconsole serial cable must be connected to serial port 0 and to node 0. An optional modem may be connected to serial port 1.

3 Power-On Self Test

POST is the Power On Self Test firmware for the V-Class platform. POST provides processor and system hardware initialization functionality, as well as providing basic processor selftest and utilities board SRAM pattern test capability. This chapter describes how POST initializes a node and handles power up errors.

Overview

Upon power up, all processors and hardware must be initialized before the node proceeds with booting. POST begins executing and brings up the node from an indeterminate state and then calls OBP.

None of the POST modules can be directly controlled via a user interface. Program control is provided by a set of configuration parameters (processing flags and variable definitions) stored in NVRAM by OBP, `do_rest`, or `xconfig`.

The error reporting modules display error codes for all fatal errors that occur during the POST execution. Any errors that can be recovered from, are reported to OBP. POST status is reflected on the LCD display.

POST performs the following tasks:

- Initializes and conditionally performs cache tests on each processor in the node
- Validates all shared data structures within the NVRAM.
- Initializes the core logic required to start OBP execution
- Determines node configuration
- Initializes all ASICs
- Initializes main memory
- Sets up CTI cache
- Invokes OBP or the Test Controller.

Any fatal errors are reported to the user by way of the system LCD and the system console. POST passes node configuration and any options to OBP via shared data structures.

Reset

The following types of reset invoke POST:

- Power up reset— If a client had execution control before the power down condition, it invokes POST to initialize the hardware. POST initializes all hardware after a power up reset.

- Hard reset—If a client had execution control before the hard reset, it invokes POST to initialize the hardware. POST restarts execution and reinitializes all hardware.
- Soft reset—If a soft reset condition has occurred while POST was executing, POST restarts execution but does not initialize main memory. It invokdes its interactive prompt.

POST modules

POST executes modules listed below in chronological order:

- Processor Initialization and Selftest—Each processor initializes itself on power up or reset in parallel with the other processors. Initialization includes setting values into the internal diagnostic registers, initializing the instruction and data caches, clearing a scratch ram area for stack and data storage, and enabling high-priority machine checks (HPMC), low-priority machine checks (LPMC), and transfer of control (TOC). Selftest includes instruction set tests, instruction and data cache RAM tests and TLB RAM tests.
- SCUB Hardware Initialization—POST clears any error state in the SCUB, initializes the SCUB hardware registers and DUART, and initializes and optionally tests the SRAM on the SCUB (see scuba_test_enable).
- Non-volatile Configuration Data Verification—POST verifies the checksum of all shared data regions in a battery-backed-up SRAM (NVRAM). POST verifies only the regions it shares with other modules, such as OBP, and those private to POST. If a region fails, it is rebuilt using default values.
- Hardware Configuration Determination—POST determines the ASIC installations status and verifies that each installed ASIC responds to register accesses. If one does not, it is reported as failing. POST then configures the system to utilize the maximum amount of installed hardware based on the V2500 hardware configuration rules.
- Node Hardware (ASIC) Initialization—POST sets up all available hardware with the proper operating mode(s) enabled. Routing is configured for the current hardware population.
- Node Main Memory Initialization—POST probes all installed memory boards for memory installation status. It then enables each memory board as a 2-, 4-, or 8-board configuration based on V2500 configuration rules. All remaining memory boards are configured to have the same logical memory population. It then initializes main memory in parallel, using up to eight processors using initialization hardware in the memory controllers.

- Page Deallocation Table Support—POST supports reading the page deallocation table (PDT) and remapping memory if it detects a bad page in the HPUX good-memory region. It updates all entries to reflect the new memory layout if remapping occurs. It also clears PDT if memory hardware change is detected.
- Client Boot—POST cleans up any residual state from POST execution and boots the client specified in `boot_module`. POST can boot clients with all processors or with just with the monarch processor leaving the other processors in an idle loop.

Interactive mode

POST for the V2500 provides a command line interface for configuration and debugging. The command line interface is invoked if boot_module is set to “interactive,” by a soft reset, or a TOC during POST execution.

Interactive mode commands

POST supports the following commands at the line prompt:

- `help`—Displays a list of supported commands and their usage.
- `banner`—Displays the POST version and build information.
- `reset [loader|post|soft]`—Causes POST to perform a reset of the node. If `loader` is specified, then the node is hard reset and executes the firmware loader PDCFL. If `post` is specified, the node is hard reset and executes POST. If `soft` is specified, the node is soft reset and executes POST.
- `dcm`—Dumps the configuration map from NVRAM and display the hardware status of the machine, showing what hardware is enabled, deconfigured, or failing.
- `setenv <parm> <value>`—Sets the configuration parameter specified by `parm` to the `value`.
- `printenv [parm]`—Prints the value of the configuration parameter specified by `parm`. If no parameter is specified, then all are printed.
- `get_opt [asic_type [asic_number]]`—Dumps the option mode bits for the ASIC type specified by `asic_type`. If an `asic_number` is also specified, then only the values of the ASIC are printed. If an `asic_number` is not specified, then all ASICs of that `asic_type` are dumped. If no `asic_type` is specified, then all ASICs are dumped.
- `pdt`—Dumps the current Page Deallocation Table (PDT) contents.
- `clear_pdt`—Clears out all entries in the PDT.
- `memmap`—Displays any rows that have been logically remapped due to PDT entries or failing software-deconfigured DIMMs.

Configuration parameters

The following parameters control the runtime operation of POST:

- `ts_ip`—Specifies the teststation IP address for LAN messaging. The value should be set to the IP address of the diagnostics LAN port on the teststation. [default: 15.99.111.99]

Table 7 Name of teststation IP address for listed utilities

Utility	Parameter name
OBP	ts-ip#
POST	ts_ip
`sppdsh`	ts_ip

- `scub_ip`—Specifies the IP address used for LAN interface hardware on the utilities (SCUB) board. This is the IP address that POST, OBP, and the Test Controller use for LAN messaging with the teststation. [default: none]

Table 8 Name of scub IP address for listed utilities

Utility	Parameter name
OBP	obp-ip#
POST	scub-ip
`sppdsh`	scub_ip
`ts_config`	scub_ip

- `cti_cache_size`—Specifies the amount of memory, in megabytes, to reserve in the node for CTI cache. This is used only in multinode configurations. [default: 0 Mbytes]

Table 9 **Name of CTI cache size IP address for listed utilities**

Utility	Parameter name
OBP	cti-cache-size
POST	cti_cache_size
sppdsh	cti_cache_size

- `boot_module`—Specifies which client to turn execution control over to at the completion of POST execution. [default: OBP]

Table 10 **Name of boot module for listed utilities**

Utility	Parameter name
OBP	boot-module
POST	boot_module
sppdsh	boot_module

- `selftest_enable`—Enables selftest control if the processor selftest is executed during POST start-up execution. [default: true]

Table 11 **Name of selftest enable for listed utilities**

Utility	Parameter name
OBP	selftest?
POST	selftest_enable
sppdsh	selftest_enable

- `scuba_test_enable`—Enables Scuba test control if the SCUB SRAM is tested before initialization. This affects the processor initialization, since processors test and initialize their own stack region. It also affects the SCUB initialization and core LAN SRAM initialization steps for the monarch. [default: true]

Table 12 Name of scuba test enable for listed utilities

Utility	Parameter name
OBP	scubatest?
POST	scuba_test_enable
sppdsh	scuba_test_enable

- `master_error_enable`—Determines whether POST will enable errors or not. This is used in conjunction with `use_error_overrides` to determine how errors are enabled. [default: true]

Table 13 Name of master error enable for listed utilities

Utility	Parameter name
OBP	master-error-enable?
POST	master_error_enable
sppdsh	master_error_enable

- `use_error_overrides`—Determines if POST will use the built-in defaults for errors or the user error overrides. This is only checked if `master_error_enable` is enabled. [default: false]

Table 14 Name of use error overides for listed utilities

Utility	Parameter name
OBP	use-error-overrides?
POST	use_error_overrides
sppdsh	use_error_overrides

- `force_monarch`—Determines if POST will force the monarch selection to a specific processor. The processor is specified in `monarch_number` [default: false]

Table 15 Name of sforce monarch for listed utilities

Utility	Parameter name
OBP	force-monarch?
POST	force_monarch
sppdsh	force_monarch

- `monarch_number`—Specifies the monarch processor when `force_monarch` is enabled. [default: 0]

Table 16 Name of monarch number for listed utilities

Utility	Parameter name
OBP	monarch#
POST	monarch_number
sppdsh	monarch_number

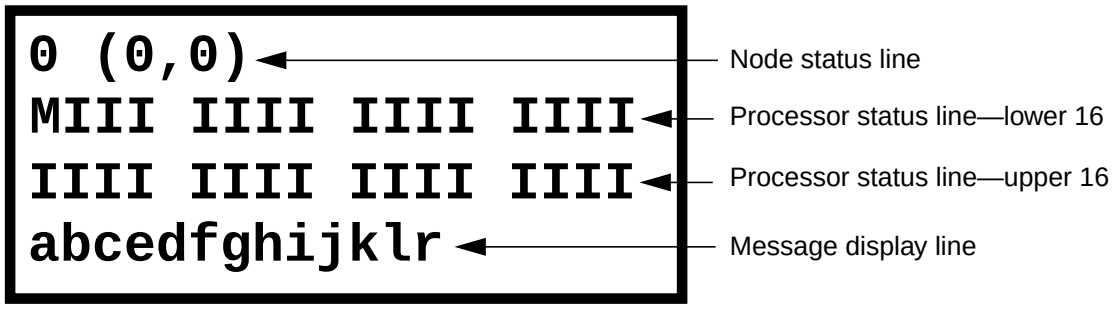
Messages

POST has three types of messages: LCD, console, and error. This section discusses each type.

LCD messages

Each node has an LCD display. Figure 29 shows the display and indicates what each line on the display means.

Figure 29 Front panel LCD



Node status line

The Node Status Line shows the physical node ID in both decimal and X, Y topology formats.

Processor status line

The processor status line shows the current run state for each processor in the node. Table 17 shows the initialization step code definitions and Table 18 shows the run-time status codes. The M in the first processor status line stands for the monarch processor.

Table 17 Processor initialization steps

Step	Description
0	Processor internal diagnostic register initialization
1	Processor early data cache initialization.
2	Processor stack SRAM test.(optional)
3	Processor stack SRAM initialization.
4	Processor BIST-based instruction cache initialization.
5	Processor BIST-based data cache initialization
6	Processor internal register final initialization.
7	Processor basic instruction set testing. (optional)
8	Processor basic instruction cache testing. (optional)
9	Processor basic data cache testing. (optional)
a	Processor basic TLB testing (optional)
b	Processor post-selftest internal register cleanup. (optional)

Table 18 Processor run-time status codes

Status	Description
R	RUN: Performing system initialization operations.
I	IDLE: Processor is in an idle loop, awaiting a command.
M	MONARCH: The main POST initialization processor.
H	HPMC: processor has detected a high priority machine check (HPMC).
T	TOC: processor has detected a transfer of control (TOC).
S	SOFT_RESET: processor has detected a soft RESET.
D	DEAD: processor has failed initialization or selftest.

Status	Description
d	DECONFIG: processor has been deconfigured by POST or the user.
-	EMPTY: Empty processor slot.
?	UNKNOWN: processor slot status in unknown.

Message display line

The message display line shows the POST initialization progress. This is updated by the monarch processor. The system console also shows detail for some of these steps. Table 19 shows the code definitions.

Table 19 Message display line

Message display code	Description
a	Utilities board (SCUB) hardware initialization.
b	Processor initialization/selftest rendezvous.
c	Utilities board (SCUB) SRAM test. (optional)
d	Utilities board (SCUB) SRAM initialization.
e	Reading Node ID and serial number.
f	Verifying non-volatile RAM (NVRAM) data structures.
g	Probing system hardware (ASICs).
h	Initializing system hardware (ASICs).
i	Probing processors.
j	Initialing, and optionally testing, remaining SCUB SRAM.
k	Probing main memory.
l	Initializing main memory.
r	Enabling system error hardware.

Console messages

POST provides several messages that are displayed on the teststation console. This section describes these console messages.

Type-of-boot

This message reports the type of boot for the current POST execution, and the node ID and monarch processor.

For example:

```
POST Hard Boot on [0:PB1R_A]
```

Version and build

This message reports the version and build information for POST.

For example:

```
HP9000/V2500 POST Release 1.0, compiled 1998/11/04
14:33:12
```

Processor probe

This message reports the processors as they are detected in the system. Only available processors are reported; any failing or deconfigured processors are not listed. Processors in this list may be deconfigured if they share a Runway bus with a processors that fails the probe or is deconfigured.

For example:

```
Probing CPUs:   PB0L_A PB0R_A PB1R_A PB1L_A PB2L_A PB2R_A PB3R_A PB3L_A
                PB4L_A PB4R_A PB5R_A PB5L_A PB6L_A PB6R_A PB7R_A PB7L_A
                PB0L_B PB0R_B PB1R_B PB1L_B PB2L_B PB2R_B PB3R_B PB3L_B
                PB4L_B PB4R_B PB5R_B PB5L_B PB6L_B PB6R_B PB7R_B PB7L_B
```

Utility board initialization

This message reports that the Utilities board SRAM reserved for missing or unavailable CPUs is being initialized. The SRAM is tested prior to initialization if `scuba_test_enable` is true.

For example:

```
Completing core logic SRAM initialization.
```

Main memory initialization

This message reports that main memory initialization has started.

For example:

```
Starting main memory initialization.
```

Memory probe

This message reports the status of the memory boards as they are detected and probed for DIMMs

For example:

```
Probing memory: MB0L MB1L MB2R MB3R MB4L MB5L MB6R MB7R
```

Installed memory

This message reports the total memory installed and available, in megabytes.

For example:

```
Installed memory: 2048 MBs, available memory: 2048 MBs
```

Main memory initialization started

This message marks the beginning of main memory initialization.

For example:

```
Initializing main memory.
```

Parallel memory initialization

This message reports that main memory initialization will be done with multiple processors in parallel. Only printed if more than one processor is available for memory initialization.

For example:

```
Parallel memory initialization in progress.
```

Memory initialization progress

This message reports the results of the initialization, the initializing processor, and the memory board for each board available in the node.

Each character indicates the physical location of the DIMM and the logical size of the DIMM. The memory information is encoded as follows:

Value	Memory Type
.	32MB
:	64MB
|	128MB
_	Empty
#	Hardware deconfigured
$	Software (user) deconfigured

For example:

```
            r0      r1      r2      r3
  PB0L_A MB0L [.... ....][.... ....][____ ____][____ ____]
  PB1R_A MB1L [.... ....][.... ....][____ ____][____ ____]
  PB2L_A MB2R [.... ....][.... ....][____ ____][____ ____]
  PB3R_A MB3R [.... ....][.... ....][____ ____][____ ____]
  PB4L_A MB4L [.... ....][.... ....][____ ____][____ ____]
  PB5R_A MB5L [.... ....][.... ....][____ ____][____ ____]
  PB6L_A MB6R [.... ....][.... ....][____ ____][____ ____]
  PB7R_A MB7R [.... ....][.... ....][____ ____][____ ____]
```

Main memory initialization complete

This message indicates that a main memory initialization is complete.

For example:

```
Main memory initialization complete.
```

System control to boot client

This message indicates that system control is being handed off to the specified boot client.

For example, one of the following:

```
Booting OBP
Booting DIAG
Booting SPSDV
Booting RDR dumper
```

```
Booting Boombox
```

Interactive boot

This message indicates that POST is entering it's interactive mode. POST provides a console interface for system configuration and debug.

For example:

```
Booting Interactive
```

Interactive prompt

The following is the POST interactive prompt and is only seen if `boot_module` is set to interactive.

For example:

```
[node id 0:monarch] POST>
```

Chassis codes

The processor initialization and selftest functions in POST report status and error information with chassis codes. These chassis codes are shared with `cpu3000` and are documented in the man page with the exception of the following POST-specific codes:

0x6103C	The processor is executing it's processor initialization code
0x22025	The processor encountered a data error while loading the processor Icache
0x22026	The processor encountered a tag error while loading the processor Icache.

Error messages

POST provides error message that are printed to the console. This section describes these error message

Teststation parameters failure

This message reports the that test station parameters structure failed

the checksum and was rebuilt to the default structure.

For example:

```
Test Station Parameters checksum FAILED, rebuilding...
```

This node may be forced with the `sppdsh reboot <node>` default command

Configuration map failure

This message indicates that the configuration map structure failed the checksum and was rebuilt to defaults. Any user deconfigured hardware state is lost.

For example:

```
Configuration Map checksum FAILED, rebuilding...
```

Configuration map failure

This message indicates that the configuration parameters structure failed the checksum and was rebuilt to the default structure. Any user overrides from the default value, for parameters that have a default, is lost. Some parameters have no default and retain the value in NVRAM. Since NVRAM could be corrupt, these values could be invalid.

For example:

```
Configuration Parameters checksum FAILED, rebuilding...
```

ASIC probe failure

This message indicates that the specified ASIC failed the probe. The status of any components that must be accessed through this component are unknown, and they are be available if installed.

For example:

```
Failed probe of P1R.
Unable to determine status of PB1R_A PB1L_A PB1L_B
PB1L_A IOLR_B.
```

Memory board deconfiguration

This message indicates that the specified memory board is deconfigured. This can be due to a memory board being found on one side of memory without a corresponding pair, since boards must be used in pairs of even/odd boards. This can also occur when a memory board has no usable memory.

For example:

Deconfiguring: MB5L

Illegal memory board configuration

This message indicates that there is an unallowed memory board configuration. Memory boards can only be used in two-, four-, or eight-board configurations. In the following example, a six-board configuration was detected, and two boards will be deconfigured.

For example:

```
Illegal 6 memory board configuration.
```

Memory remap

This message indicates that the specified physical row was mapped to the indicated logical row. This is done to accommodate either improperly installed DIMMs or when DIMMs in lower rows are not usable. This can occur when a DIMM is bad or when memory boards contain differing memory populations.

For example:

```
MB0L: Physical Row:2 mapped to Logical Row:0
```

Processor initialization failure

This message indicates that the specified processor failed to perform the step described during parallel main memory initialization. The monarch processor completes the initialization assigned the failing processor.

For example:

```
PB1R_A timed out during encache memory init code
PB1R_A timed out during memory initialization
PB1R_A timed out during idle request after memory init
```

```
PB0L_B failed to go idle after memory init
Unable to force CPU PB2L_A into idle loop
```

Monarch completing memory initialization

This message indicates that the monarch processor is completing the memory initialization assigned to the specified processor.

For example:

```
Using Monarch to initialize memory assigned to PB2L_A
```

PDT checksum failure

This message indicates that the page deallocation table structure failed the checksum and was rebuilt to defaults. All bad page information is lost.

For example:

```
Page Deallocation Table (PDT) checksum FAILED,
rebuilding...
```

Memory hardware change detected

This message indicates that POST detected a change in memory hardware and cleared all entries in the PDT.

For example:

```
Detected a hardware change, clearing the Page
Deallocation Table (PDT).
```

Memory remapped

This message indicates that POST remapped memory to achieve HP-UX good memory region. This occurs when a bad page is marked within the good memory region.

For example:

```
Memory was re-mapped to achieve HP/UX good memory
region.
```

Contiguous memory block not found

This message indicates that POST could not find a block of contiguous memory to place at address zero to achieve good memory. POST will report no main memory to the OBP for this failure.

For example:

```
HP/UX good memory region could not be achieved.
```

Processor not reported

This message indicates that a processor failed to mark itself in the system report register. Reporting happens early in the sequence, and this failure usually indicates the processor has failed to execute any instructions.

For example:

```
Failed probe of PB1R_B, CPU failed to report in.
```

Processor initialization/selftest failure

This message indicates that a processor failed at some point during initialization or selftest. The chassis code for the module that failed is reported.

For example:

```
Failed probe of PB1R_B
chassis code 0x6103C
```

Processor not responding to interrupt

This message indicates that a processor properly initialized itself but did not respond to an external interrupt

For example:

```
Failed probe of PB1R_B
cpu PB1R_B did not respond to an interrupt
```

Shared Runway bus failure

This message indicates that an available processor has been deconfigured because it shares a Runway bus with a processor that failed to probe

For example:

```
cpu PB1R_A deconfigured due to PB1R_B shutdown.
```

New monarch processor selected

This message indicates that the previous monarch processor was deconfigured and a new one was selected. The new monarch continues the initialization of the rest of the system

For example:

```
INFO: New monarch selected: PB0R_A
```

New monarch processor not found

This message indicates that the other processor on the Runway bus with the monarch processor was deconfigured or failed and another suitable processor could not be found to replace the monarch.

For example:

```
WARNING: The monarch shares a Runway bus with a failed
cpu.
```

4 Test Controller

The Test Controller is an EEPROM-based utility that provides the environment for executing the offline diagnostic tests. It is controlled through parameters stored in the NVRAM on the Utilities board. The Test Controller reads these parameters to determine its execution mode, the number processors to test, which SMACs to include in the testing, which subtests to run, and other diagnostic test-specific information.

Test Controller modes

There are three basic operational modes for this utility:

- Stand-alone mode
- Interactive mode
- I/O Utility mode

In stand-alone mode, `cxtest` invokes the Test Controller. The Test Controller reads test parameters from NVRAM (these parameters are written into NVRAM by `cxtest` before it invokes the Test Controller), executes the test and subtests specified in NVRAM, and sets a completion bit in NVRAM when the test and subtests are finished. `cxtest` is described in Chapter 5.

In interactive mode, a user interface allows the user to select the processors to test, select the subtests to run, and examine error information. The user interface is a set of menus described in this chapter.

In the interactive mode, the Test Controller loops waiting for the start command. Prior to issuing the start command, any global and/or processor specific parameters can be modified. When all tests have completed, the Test Controller waits for the next start command. Any combination of parameter and tests may be modified and executed.

In I/O Utility mode, the Test Controller will load, and subsequenty exectue, a firmware utility module from the test station. There are currently two support utility modules: `arrm` and `dfdutil`. The teststation utility `tc_ioutil` identifies the utility module to be loaded. `tc_ioutil` updates an NVRAM location with the file name of the utility module to be loaded. See `tc_ioutil`, `arrm` and `dfdutil` in Chapter 11, “Utilities,” or more details.

An example of tc_ioutil would be:

```
tc_ioutil <node> dfdutil.fw
```

User interface

The Test Controller provides for the control of offline diagnostic test execution. It utilizes a set of parameters to control its operation. The parameters consist of the following:

- Global set that controls the overall operation of the Test Controller
- Test set (one per test) that controls how the tests are executed by the Test Controller
- CPU parameters (one per processor) that contain status information about the tests executing on each processor

All these parameters are in NVRAM.

The user interface allows the user to modify parameters that reside in NVRAM, thereby controlling the operation of the Test Controller. It also allows the user to select which subtests are executed on each of the processors and modify the test parameters, as well as any other test information.

The Test Controller user interface consists of two basic menus. The first is the main menu that gives the user the following capabilities:

- Modify the POST boot selection
- Control operation of the Test Controller
- Display the current global parameter selections
- Display processor summary
- Switch processors
- Go to the Test Configuration menu

The second menu is the processor Test Control menu that provides the following capabilities:

- Select classes of subtests to execute
- Select subtests to execute
- Specify pause enables
- Specify whether or not to loop
- Specify the test and/or subtest error counts

- Read and write the 128 words of test specific information
- Select the hardware to test
- Display the current parameter selections

Main menu

Test Controller Main Menu

```
MAIN Menu commands

     0=Quit Test Controller
     1=Begin Test Controller Execution
     2=Halt Test Controller Execution
     3=Resume Test Controller Execution
     4=Switch CPU
     5=POST Boot Selection
     6=Execution Mode Selection
     7=Global Parameter Display
     8=CPU Summary Display
     9=Display CPU Errors
     A=Test Selection Menu
     B=Test Configuration Menu
     C=Debugging Menu
     D=Display revision

Enter Command:
```

Each main menu selection is defined as follows:

- 0=Quit Test Controller—Terminates the Test Controller utility and either reboots the system (to POST and then to the selected program) or halts the system depending on the current value of the POST Boot Selection flag.
- 1=Begin Test Controller Execution—Starts the Test Controller utility executing the specified subtests on the selected processors. The entire system is started from the beginning.
- 2=Halt Test Controller Execution—Suspends temporarily the operation of the Test Controller. This command may be entered at any time. Only the Test Controller is halted; subtests on other processors continue to execute.

- 3=Resume Test Controller Execution—Continues execution from the point of interruption.
- 4=Switch CPU—Allows the user to start the Test Controller on the specified processor. The previously used processor starts executing the command wait loop code. The user is prompted for the processor as follows:

  Enter CPU (0-1f):

- 5=POST Boot Selection—Prompts the user for the new value with the following prompt:

  Boot Option (1=OBP, 2=TC Interactive, 3=TC Standalone
  4=Loader, 5=SPSDV, 6=RDR Dumper,
  7=Boombox, 8=POST Interactive):

  For all values, POST boots to ESPDV. The Test Controller performs a hard reset to POST when the Test Controller terminates.

- 6=Execution Mode Selection—Allows the user to select the mode for executing the subtests. The two options are serial and parallel. The following prompt queries for the selection:

  ```
  Execution Mode Selection (0=serial, 1=parallel):
  ```

- 7=Global Parameter Display—Displays the available hardware components, the current “POST Boot Selection” value, and the current “Execution Mode Selection” value. The display is shown below:

Example Global Parameter display

```
Enter command: 7

MAIN Menu - Global Parameters Display
    CPUs:                0* 1* 2* 3* 4* 5* 6* 7* 8* 9* A* B* C* D* E* F*

                        10*11*12*13*14*15 16*17*18*19*1A*1B*1C*1D 1E*1F
    SPACs:               0* 1* 2* 3* 4* 5* 6* 7*
    SMACs:               0* 1* 2  3  4  5  6* 7*
    STACs:               0  1  2  3  4  5  6  7
    SAGAs:               0  1* 2  3  4  5* 6  7
    Execution Mode Selection:  Serial  Parallel*
    POST Boot Selection:       TC Interactive
```

The asterisks denote the component has passed POST processing and is available for diagnostic testing (see processors 0-3 and SMACs 0, 2, 4, and 6 in the display).

- 8=CPU Summary display—Displays a summary of the current processor and testing information. An example of the display is shown below:

Example CPU summary display

```
MAIN Menu - CPU Summary Display
    Total Failures = 0
  Configuration Map
  =================
  CPUs :  0   1   2   3*  4   5   6   7   8   9  10  11  12  13  14  15
  CPUs : 16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31
 SPACs :  0*  1*  2*  3*  4*  5*  6*  7*
 SMACs :  0*  1*  2   3   4   5   6   7
 STACs :  0   1   2   3   4   5   6   7
 SAGAs :  0*  1   2   3   4*  5   6   7

                    FAIL
  CPU STATE         COUNT SUBTEST TEST NAME
  === =====         ===== ======= =========
    0 Not Available n/a   n/a     n/a
    1 Not Available n/a   n/a     n/a
    2 Not Available n/a   n/a     n/a
    3 Idle          n/a   n/a     n/a
    4 Not Available n/a   n/a     n/a
    5 Not Available n/a   n/a     n/a
    6 Not Available n/a   n/a     n/a
    7 Not Available n/a   n/a     n/a
    8 Not Available n/a   n/a     n/a
    9 Not Available n/a   n/a     n/a
   10 Not Available n/a   n/a     n/a
   11 Not Available n/a   n/a     n/a
   12 Not Available n/a   n/a     n/a
   13 Not Available n/a   n/a     n/a
   14 Not Available n/a   n/a     n/a
   15 Not Available n/a   n/a     n/a
   16 Not Available n/a   n/a     n/a
   17 Not Available n/a   n/a     n/a
   18 Not Available n/a   n/a     n/a
   19 Not Available n/a   n/a     n/a
   20 Not Available n/a   n/a     n/a
   21 Not Available n/a   n/a     n/a
   22 Not Available n/a   n/a     n/a
   23 Idle          n/a   n/a     n/a
   24 Idle          n/a   n/a     n/a
   25 Not Available n/a   n/a     n/a
   26 Not Available n/a   n/a     n/a
   27 Not Available n/a   n/a     n/a
   28 Not Available n/a   n/a     n/a
   29 Not Available n/a   n/a     n/a
   30 Not Available n/a   n/a     n/a
   31 Not Available n/a   n/a     n/a
Hit <ENTER> key to return to the MAIN Menu:
```

Each available hardware component is marked with an asterisk just to the right of its number (see processors 0-3 and SMACs 0, 2, 4, and 6 in the display).

The possible states in the CPU Summary Display are described in Table 20.

Table 20 Processor States

CPU State	Description
Not Available	Denotes processor is not available for testing.
Running	Denotes a test is currently running on this processor.
Idle	Denotes that no test is running on this processor.
Ready	Denotes last subtest completed and ready for next subtest.
Test Completed	Denotes test completed execution on this processor.
Error Detected	Denotes test halted due to an error condition on this processor.
Test Timeout	Denotes a timeout detected during test execution on this processor; the test is halted.
HW Reqs Not Met	Denotes the hardware selected does not meet the minimum hardware required for executing the test.
User Halted	Denotes user halted test.
Unexpected HPMC	Denotes running test caused an HPMC; the test is halted.
SW Deconfigured	Denotes test automatically halted testing on this processor, because of a software restriction.

- 9=Display CPU Errors—Displays the errors for the currently selected processor. When selected, the user is prompted for the processor as follows:

  ```
  Enter CPU [0-1f]:
  ```

Example Test Parameters display.

```
Test Configuration Menu - Test Parameters Display
  CPUs:  ( 1)  0  1  2  3* 4  5  6  7  8  9  A  B  C  D  E  F
              10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
  SPACs: ( 1)  0* 1* 2* 3* 4* 5* 6* 7*
  SMACs: ( 0)  0* 1* 2  3  4  5  6  7
  STACs: ( 0)  0  1  2  3  4  5  6  7
  SAGAs: ( 0)  0* 1  2  3  4* 5  6  7
  Nodes: ( 1)
  Loop Enable:            ON   OFF*
  Loop Count:             00
  Test Error Count:       01
  Pause Test Start:       ON   OFF*
  Pause Test End:         ON   OFF*
  Pause Subtest Start:    ON   OFF*
  Pause Subtest End:      ON   OFF*
  Pause On Fail:          ON*  OFF
```

There are four fields:

- Date/Time—Date and time the error was logged.
- Subtest—Failing subtest number.
- Event Code—32-bit event code.
- Error Message—40-character error message

- A=Test Selection Menu—Invokes the menu. This menu allows the user to select which tests to execute. An asterisk before the test name denotes that it has been selected. Only the memory test is selected. Selecting options 1 through A toggles the current state for that particular test. Selecting option 0 returns the user to the main menu shown below:

Test Selection display

```
MAIN Menu - Test Selection Display

0=Return to Main Menu
1=*Memory test
2=not available
3=not available
4=not available
5=I/O test
6=CPU selftests
7=not available
8=not available
9=not available
A=not available
Please enter number of test:
```

- B=Test Configuration Menu—Switches the user to the Configuration menu shown below for the specified test.
- C=Debugging Menu—Invokes the Debugging Menu shown below that allows the user to read or write to any memory location on the node and dump various data.

Example Debugging menu

```
MAIN Menu - Debugging Menu

0=Return to Main Menu
1=Read 32-bit Memory Location
2=Write 32-bit Memory Location
3=Read 64-bit Memory Location
4=Write 64-bit Memory Location
5=Read ECC from memory line
6=Read Tag From Memory Line
7=Print Test Revision
8=Dump last HPMC Info
9=Clear All Error Info
A=Dump TC CPU Info Structure
B=Dump TC Test Info Structure
C=Reset SONIC Interface
D=Dump SONIC Registers
Enter number of activity:
```

- Selection 1 queries for the 40-bit address to read as follows:

  ```
  Enter 40-bit address:
  ```

- Selection 2 queries for the 40-bit address and then for the 32-bits of data to write:

  ```
  Enter 32-bit data:
  ```

- Selection 3 queries for the 40-bit address to read.
- Selection 4 queries for the 40-bit address to write, and then for the 64-bits of data to write as follows:

  ```
  Enter Upper 32-bits:
  ```

  ```
  Enter Lower 32-bits:
  ```

- Selections 5 and 6 query for the 40-bit address for which to display the ECC or tag of that memory line.
- Selection 7 queries the user for the test index with the following prompt:

  ```
  Enter test index [1-A]:
  ```

- Selection 8 prints information for the last HPMC that occurred on the specified processor.
- Selection 9 clears all stored error information.

NOTE Use caution using selection 9 as there is no undo function.

- Selection A queries the user for the processor index as follows:

  ```
  Enter cpu index [0-1f]:
  ```

- Selection B queries the user for the test index with the prompt shown.

  ```
  Enter test index [1-A]:
  ```

Test Configuration menu

The Test Configuration menu is shown below:

Test Configuration menu

```
Test Configuration Menu

0=Return to Main Menu A=Hardware Selection Menu
1=Display ClassesB=Loop Enable
2=Display SubtestsC=Loop Count
3=Select ClassesD=Test Error Count
4=Select SubtestsE=Pause at Test Start
5=Read All Test ParametersF=Pause at Test End
6=Read One Test ParameterG=Pause at Subtest Start
7=Write Test ParameterH=Pause at Subtest End
8=Reset ParametersI=Pause On Failure
9=Display Test Configuration

Enter command:
```

Each Test Configuration menu selection is defined as follows:

- 0=Return to Main Menu—Returns the user to the Main menu.
- 1=Display Classes—Displays the current class definitions for this diagnostic. An example of the display is shown below:

Test Configuration menu - Class display

```
Test Configuration Menu - Class Display

Class  Description
0         class 0  description
1         class 1  description
.              .
.              .
n         class n description
```

- 2=Display Subtests—Displays the current subtest definitions for this diagnostic. An example of the display is shown in the example belolw.

Test Configuration menu - Subtest display

```
Test Configuration Menu - Subtest Display

Subtest     Description
0           subtest 0 description
1           subtest 1 description
.            .
.            .
n*         subtest n description
```

An asterisk following the subtest number denotes that it is selected for execution. For example, see the “n subtest n description” line.

- 3=Select Classes—Allows the user to specify which classes of subtests to execute. These selections are in addition to any subtests selected. The following prompt is displayed:

  `Enter class number:`

  The user must enter one of the following:

  - An optional operator followed by a class number, for example 2, +2 or -2.
  - Multiple class numbers (class, +class, -class are allowed) separated by commas or spaces, for example 1,2,3.

    The class numbers are decimal.

- 4=Select Subtests—Allows the user to specify which subtests to execute. These selections are in addition to any classes selected. The following prompt is displayed:

  `Enter subtest number or subtest range:`

  The user may enter one of the following:

  - A single subtest number.
  - A subtest range which consists of two numbers separated with a dash and is inclusive. An example of a valid range entry is 100-200.
  - An optional operator followed by a subtest selection, for example 2, +2, -2, +100-200, or -100-200.

  The subtest numbers are decimal values only.

- 5=Read All Test Parameters—Reads all 128 words that make up the test parameter set and displays this information. These test parameters reside in NVRAM and are defined by the particular test. An example of the display is shown in the example below:

Test Configuration menu - Test Parameters

```
Test Configuration Menu - Test Parameters

WordValueWordValue
00xhhhhhhhh100xhhhhhhhh
10xhhhhhhhh110xhhhhhhhh
20xhhhhhhhh120xhhhhhhhh
30xhhhhhhhh130xhhhhhhhh
40xhhhhhhhh140xhhhhhhhh
50xhhhhhhhh150xhhhhhhhh
60xhhhhhhhh160xhhhhhhhh
70xhhhhhhhh170xhhhhhhhh
80xhhhhhhhh180xhhhhhhhh
90xhhhhhhhh190xhhhhhhhh
```

- 6=Read One Test Parameter—Allows the user to read a single test parameter.
- 7=Write Test Parameter—Allows the user to modify any one of the test parameter words. The user is first requested to specify which word is to be modified:

  ```
  Specify Test Parameter word [0-127]:
  ```

  After specifying the word, the user is then prompted for the new value:

  ```
  New value for Test Parameter word xx:
  ```

- 8=Reset Parameters—Resets some of the parameters to their default values. Table 21 lists the affected parameters and their default values.

Table 21 Parameter Defaults

Parameter	Default value
Loop Enable	0
Loop Count	0
Test Error Count	1
Pause At Test Start	0
Pause At Test End	0
Pause At Subtest Start	0
Pause At Subtest End	0

- 9=Display Test Configuration—Displays the current values of the processor parameters. An example of the display is shown in the example below. An asterisk denotes the current selections. For Example, processor 0 is selected.

  This minimum hardware requirements information is enclosed in parentheses after the hardware type label and denotes the number of that type required. In the example below, 1 processor, 1 PAC (the one associated with the selected processor), and 1 MAC are needed.

  The Test Controller compares the selected hardware components versus these minimum requirements to determine if the test can be executed. Unless these minimum requirements are met, the test cannot be executed.

Test Configuration menu - Test Parameters display

```
Test Configuration Menu - Test Parameters Display
  CPUs:  ( 1)  0  1  2  3* 4  5  6  7  8  9  A  B  C  D  E  F
               10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
  SPACs: ( 1)  0* 1* 2* 3* 4* 5* 6* 7*
  SMACs: ( 0)  0* 1* 2  3  4  5  6  7
  STACs: ( 0)  0  1  2  3  4  5  6  7
  SAGAs: ( 0)  0* 1  2  3  4* 5  6  7
  Nodes: ( 1)
  Loop Enable:            ON   OFF*
  Loop Count:             00
  Test Error Count:       01
  Pause Test Start:       ON   OFF*
  Pause Test End:         ON   OFF*
  Pause Subtest Start:    ON   OFF*
  Pause Subtest End:      ON   OFF*
  Pause On Fail:          ON*  OFF
```

A=Hardware Selection menu—Invokes Hardware Selection menu shown in the example below:

Test Configuration menu - Hardware Selection menu

```
Test Configuration Menu - Hardware Selection Display

0=Return to Test Configuration Menu
1=CPU Selection
2=SPAC Selection
3=SMAC Selection
4=STAC Selection
5=SAGA Selection
6=Node Selection
Enter Command:
```

The selections of the Hardware Selection menu are defined as follows:

- 1-5=<hardware> Selection—Selects the appropriate controller. The following prompt is displayed:

  `Select <hardware>:`

  The user must enter one of the following:

  - An optional operator followed by a hardware component number, for example 2, +2 or -2.

- Multiple hardware component numbers separated by commas or spaces, for example 1,+2,-3.

The format 2, or +2, denotes to use this hardware component in testing. The format -2 denotes not to use this hardware component in testing. The 1 and +1 formats are equivalent, and leaving a hardware component out of the list is equivalent to the -n format.

As an example, to select all the even processors the user would enter:

0,2,4,6,8,a,c,e, 10, 12, 14, 16, 18, 1a, 1c, 1e

- 6=Node Selection—Allows the user to enter 7-bit node ids. The format is similar to that used for processors, i.e. an optional operator can be used and multiple entries are allowed. The following prompt is used:

```
Enter Node Ids:
```

- B=Loop Enable—Allows the user to modify the value of the loop enable flag, which causes the Test Controller utility to loop on all selected subtests when the last subtest is executed. The user is prompted for the new value as follows:

```
Loop Enable (0=disabled, 1=enabled):
```

- C=Loop Count—Allows the user to specify the number of times to loop through the selected subtests. It is only used if the “Loop Enable” flag is set. The user is prompted for the new value (a decimal number) as follows:

```
Loop Count:
```

- D=Test Error Count—Allows the user to modify the maximum number of test errors that can occur before the Test Controller utility terminates execution of subtests on this processor. The user is prompted for the new value (a decimal number) as follows:

```
Test Error Count value (1-127, N=no limit):
```

A value of N means that there is no limit to the number of errors that can occur.

- E=Pause at Test Start—Allows the user to modify the pause at test start flag. This flag results in the Test Controller pausing the testing on this processor just prior to starting the process of determining the first subtest to execute. The user is prompted for the new value as follows:

```
Pause at Test Start (0=disabled, 1=enabled):
```

- F=Pause at Test End—Allows the user to modify the pause at test end flag. This flag results in the Test Controller pausing the testing on this processor after last subtest has completed execution and all cleanup is complete. The user is prompted for the new value as follows:

  ```
  Pause at Test End (0=disabled, 1=enabled):
  ```

- G=Pause at Subtest Start—Allows the user to modify the pause at subtest start flag. This flag results in the Test Controller pausing the testing on this processor just prior to starting the execution of the current subtest. The user is prompted for the new value as follows:

  ```
  Pause at Subtest Start (0=disabled, 1=enabled):
  ```

- H=Pause at Subtest End—Allows the user to modify the pause at subtest end flag. This flag results in the Test Controller pausing the testing on this processor just after detecting that the current subtest has completed execution. The user is prompted for the new value as follows:

  ```
  Pause at Subtest End (0=disabled, 1=enabled):
  ```

- I=Pause on Failure—Allows the user to specify whether testing should halt when the specified number of failures is detected. The default is to halt. The user is prompted for the new value as follows:

  ```
  Pause in Failure (0=disabled, 1=enabled)
  ```

Example of running diagnostics from Test Controller command line

This example shows how to run `mem3000` from the Test Controller command line within the following scenario:

- Configure `mem3000` to run on a system with four memory boards installed.
- Set the classes and subtests to be executed.
- Run the tests.
- View the results.

Configuration

To execute the scenario, perform the procedures in this and the following sections:

Step 1. From the Test Controller main menu shown below, select the Test Selection Menu, option A:

Test Controller main menu

```
MAIN Menu commands

0=Quit Test Controller
1=Begin Test Controller Execution
2=Halt Test Controller Execution
3=Resume Test Controller Execution
4=Switch CPU
5=POST Boot Selection
6=Execution Mode Selection
7=Global Parameter Display
8=CPU Summary Display
9=Display CPU Errors
A=Test Selection Menu
B=Test Configuration Menu
C=Debugging Menu
D=Display revision
```

Step 2. From the Test Selection menu shown below, select Memory test, option 1.

Test Controller Test Selection menu

```
MAIN Menu - Test Selection Display

    0= Return to Main Menu
    1= Memory test
    2= not available
    3= not available
    4= not available
    5= I/O test
    6= CPU Selftests
    7= not available
    8= not available
    9= not available
    A= not available
Please enter number of test:
```

Step 3. Select option 0 to return to the Main Menu

Step 4. Select option B, Test Configuration Menu, from the Main Menu.

Step 5. From the menu, select Memory test, option 1.

This opens the Test Configuration menu shown below:

Test menu

```
1=*Memory test
2= not available
3= not available
4= not available
5= I/O test
6= CPU Selftests
7= not available
8= not available
9= not available
A= not available
Please enter number of test:
```

Step 6. From the Test Configuration menu shown below, select the Hardware Selection menu, option A.

Test Controller Test Selection menu

```
Test Configuration Menu

     0=Return to Main Menu          A=Hardware Selection Menu
     1=Display Classes              B=Loop Enable
     2=Display Subtests             C=Loop Count
     3=Select Classes               D=Test Error Count
     4=Select Subtests              E=Pause at Test Start
     5=Read All Test Parameters     F=Pause at Test End
     6=Read One Test Parameter      G=Pause at Subtest Start
     7=Write Test Parameter         H=Pause at Subtest End
     8=Reset Parameters             I=Pause On Failure
     9=Display Test Configuration

Enter command:
```

Step 7. From the Hardware Selection menu shown below, select CPUs, option 1.

Selecting CPUs from Hardware Selection menu

```
Test Configuration Menu - Hardware Selection Display
   0=Return to Test Configuration Menu
   1=CPU Selection
   2=SPAC Selection
   3=SMAC Selection
   4=STAC Selection
   5=SAGA Selection
   6=Node Selection
```

Step 8. At the following prompt:

```
Select CPUs: 0 2
```

Select the number of processors (CPUs).

After the number of processors is chosen, the Hardware Selection menu reappears.

Step 9. From this menu select Return to Test Configuration menu, option 0.

Step 10. From the Test Configuration menu, the user can select option 9 to view the changes.

Selecting classes and subtests

To select test classes and subtest, perform the following procedure:

Step 1. From the Test Configuration menu, select Select Classes, option 3.

Step 2. From the following prompt, select the test classes:

```
Enter class number:
```

Step 3. From the Test Configuration menu, select Display Subtests, option 2.

The subtest menu shown below lists all available subtests:

mem3000 Subtests menu

```
100*   Diagnostic CSR Read/Write Test
101*   Other SMAC CSR Read/Write Test
110*   Memory Data Read/Write Test
120*   Memory ECC Read/Write Test
130*   Memory Tag Read/Write Test
140*   Memory Initialization Test
150*   First 32 Memory Lines Test
200*   Tag Bank Test
210*   Tag Addressing Test
211*   Tag Byte Uniqueness Pattern Test
230*   Tag March-C Pattern #1 Test
231*   Tag March-C Pattern #2 Test
232*   Tag March-C Pattern #3 Test
233*   Tag March-C Pattern #4 Test
234*   Tag March-C Pattern #5 Test
235*   Tag March-C Pattern #6 Test
236*   Tag March-C Pattern #7 Test
237*   Tag March-C Pattern #8 Test
238*   Tag March-C User Data Pattern Test
300*   Memory Bank Test
310*   Memory Addressing Test
311*   Byte Uniqueness Pattern Test
330*   Memory March-C Pattern #1 Test
331*   Memory March-C Pattern #2 Test
332*   Memory March-C Pattern #3 Test
333*   Memory March-C Pattern #4 Test
334*   Memory March-C Pattern #5 Test
335*   Memory March-C Pattern #6 Test
336*   Memory March-C Pattern #7 Test
337*   Memory March-C Pattern #8 Test
338*   Memory March-C User Data Pattern Test
400*   Memory Load/Store Test
410*   Memory Data Flush Transaction Test
420*   Memory Semaphore Transaction Test
500*   TAG ECC Single Error Correction Test
501    DATA ECC Single Error Correction Test
502    ECC Single Error Correction Test
510*   ECC Double Error Detection (coherent) test
520*   ECC Double Error Detection (non-coherent) test
530*   ECC Disable Test
600*   Memory Access Protection Test
610*   Memory Tag Test I
640*   80 vs. 88 Bit DIMM Test
```

Step 4. Select all appropriate subtests. Table 22 lists the test patterns for subtests 230 through 238.

Table 22 Test patterns for subtests 230-238 and 330-338

Subtest	Patterns
230, 330	0x7f7f7f7f7f7f7f7f 0x8080808080808080
231, 331	0xbfbfbfbfbfbfbfbf 0x4040404040404040
232, 332	0xdfdfdfdfdfdfdfdf 0x2020202020202020
233, 333	0xefefefefefefefef 0x1010101010101010
234, 334	0xf7f7f7f7f7f7f7f7 0x0808080808080808
235, 335	0xfbfbfbfbfbfbfbfb 0x0404040404040404
236, 336	0xfdfdfdfdfdfdfdfd 0x0202020202020202
237, 337	0xfefefefefefefefe 0x0101010101010101
238, 338	0xa5a5a5a5a5a5a5a5 0x5a5a5a5a5a5a5a5a

Selecting Display Subtests, option 2, from the Test Configuration Menu reflects the changes.

Starting tests

To run the tests selected from the Test Controller main menu, select Begin Test Controller Execution, option 1. The output is shown in the example below:

Example of `mem3000` execution

```
% Enter command: 1

 Execution Starting.
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................

 Execution Completed - No errors detected
```

Viewing the results

To review the results of the test, select CPU Summary Display, option 8, from the Main menu.

An example of the results is shown below:

Example CPU Summary diaplay

```
  Configuration Map
  =================
  CPUs :  0   1   2   3*  4   5   6   7   8   9  10  11  12  13  14  15
  CPUs : 16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31
 SPACs :  0*  1*  2*  3*  4*  5*  6*  7*
 SMACs :  0*  1*  2   3   4   5   6   7
 STACs :  0   1   2   3   4   5   6   7
 SAGAs :  0*  1   2   3   4*  5   6   7
                      FAIL
  CPU STATE           COUNT SUBTEST           TEST NAME
  === =====           ===== =======           =========
  0  Test Completed   0     150     MEM3000 - EEPROM based memory tests
  1  Idle             n/a   n/a    n/a
  2  Test Completed   0     150     MEM3000 - EEPROM based memory tests
```

5 cxtest

The `cxtest` program is a graphical front end and a command line interpreter for the test controller. It is a standalone program that runs independently of any diagnostic tests loaded in the EEPROM on the Utilities board.

Overview

The `cxtest` program runs on the teststation and communicates with the test controller via the NVRAM configuration parameters on the Utilities board. Depending on the command line, `cxtest` either starts the graphics display or runs as a command line interpreter.

The GUI provides an easy and flexible way to select and run tests. The main screen has six drop down menus. The six menus are, File Menu, Test Menu, Global Parm Menu, Command Menu, System Configuration Menu, and Help Menu. The menus contents are present in the sections below.

The advantage of using the command line over the graphics user interface is that there is a little greater control over the order in which tests are run and the ability to run the tests from a script. The disadvantage is that the user must be more aware of what tests are installed and how to run them.

The test controller must be in the stand-alone mode in order for `cxtest` to be able to communicate with it. To run the test controller in stand-alone mode, run the following command:

```
do_reset <node> | tc_standalone
```

For example:

```
do_reset 0 | tc_standalone
```

When `cxtest` is invoked, it first retrieves system information from NVRAM and EEPROM on the Utilities board. This information includes:

- Tests loaded
- Parameters required for those tests
- Hardware configuration

The `cxtest` program works with the test controller to execute tests based on the options selected by the user. It performs the following functions:

- Looping
- Dispatching tests
- Configuring hardware

- Retrieving error information from the test controller

The test controller operates in the standalone mode when running in conjunction with `cxtest`. This is true whether one is using the command line version of `cxtest` or the graphics interface.

Graphics interface

To start the `cxtest` graphics interface, specify the `-d` option on the command line as follows:

```
% cxtest -d
```

This causes `cxtest` to open a window on the display. Where the window is displayed is set by the environment variable $DISPLAY. This cannot be changed on the command line.

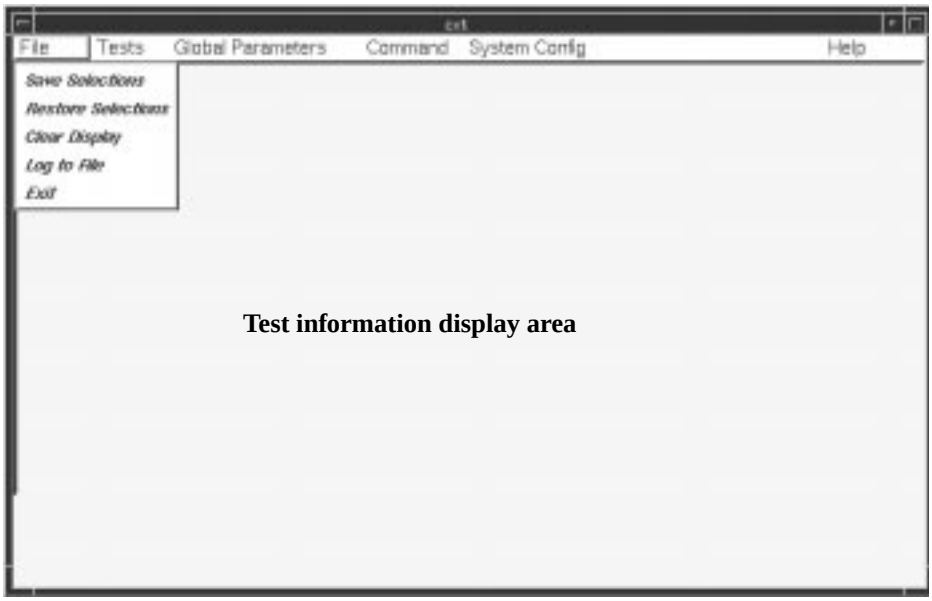
The window has two areas of importance:

- Menu selections
- Test information display

Menus

There are six main menus in `cxtest`. Figure 30 shows the cxtest menu bar.

Figure 30 **cxtest menu**



File menu

The File menu has the following options:

- Save Selections
- Restore Selections
- Log to File/Close Log File
- Clear Log
- Exit

Save Selections

The Save Selections option saves specific tests or configurations.

Restore Selections

With the Restore Selections option, the user runs specific tests without having to click on many buttons.

Clear Display

This option clears the browser of all text. It does not clear the log file.

Log to File/Close Log File

This option starts logging the information to the file /spp/data/<COMPLEX_NAME>/cxtest.log. The previous file is not saved. No information present on the screen prior to this option being enabled is saved.

Exit

The Exit option closes `cxtest`.

When exiting `cxtest`, the state of the Boot option is set to what is displayed in the System Configuration menu. The default is to return to OBP, so if the user intends to return to `cxtest`, make sure the test controller stand-alone option is checked. See Figure 33.

Test menu

Selecting a test from the Test menu opens a window like the one in Figure 31. The Test menu varies depending on the tests loaded in EEPROM. If there is only one test loaded in EEPROM, then only one test appears in this window. The test names are presented as they appear in the EEPROM. If a test is not present in EEPROM, there is a “-” in the menu.

The selections presented are based on whether the Test Controller has built a Subtest table and Class table in its `tc_test_info_struct` structure.

Class menus

Selecting a test opens a window that displays all classes for the test. See Figure 31. Down the left hand side of the window are a column of round buttons, and down the right hand side of the window are two columns of buttons.

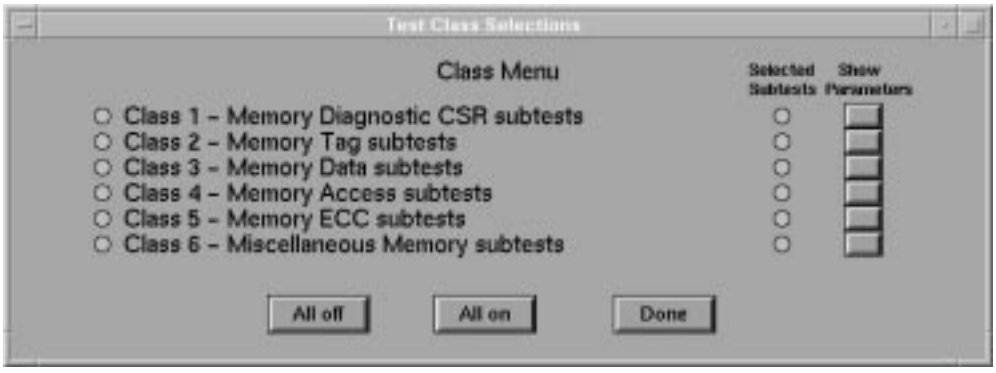
Subtest menus

The left column of buttons allows the user to select all the subtests in the class with a single click. The button will turn yellow when selected.

The round buttons on the right hand side select subtests (opening another window for theses choices) within that class. If only some of the subtests in a class are selected, then both the round buttons for that class turn yellow.

Consider the round button on the left as an indication that tests within the class have been selected and the round button on the right as an indication that only some of the tests have been selected, as opposed to all the tests of that class.

Figure 31 Test Class Selection menu



The last button on the right hand side associates parameters with the test class. Clicking this button opens another window with the parameters for that class. If there are more parameters than will fit on the screen, a scroll bar allows the user to scroll the list.

Also above and below each of the scroll bars are two buttons that allow the user to page up or down through the parameter list.

The Defaults button installs test default values into all the parameters.

If a class of tests has no parameters associated with it, the right most button (the square one) is not shown.

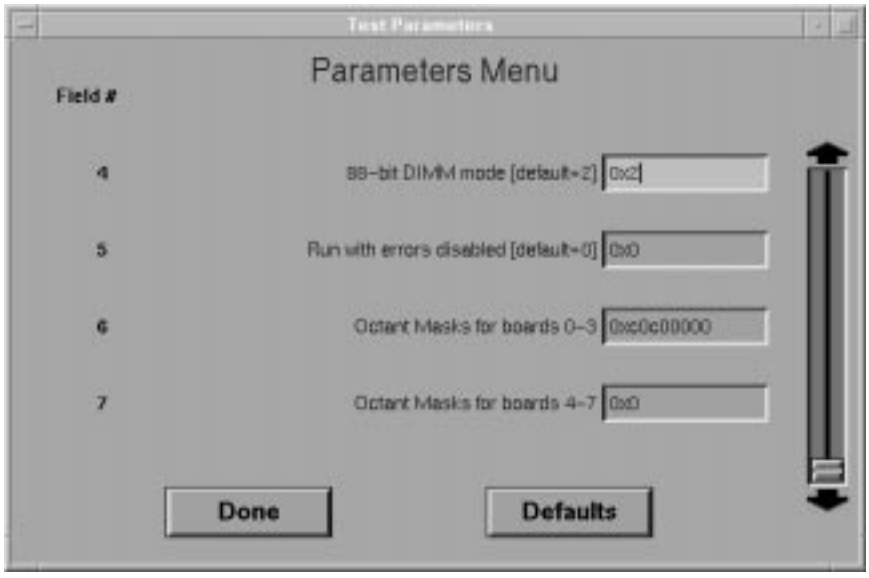
Global Test Parameters menu

`cxtest` provides the ability to loop on a number of tests by setting the Loop Enable count. The looping parameter is applied on a per test basis and is applied to all the tests. As an example it the user selects two subtests from the cpu test and one class from the memory test and sets the loop parameter to 3, then the two subtests from the cpu test are repeated three times in. Then the class of tests from the memory test are repeated three times.

A number of different pausing options are selectable. The pause can be selected before a class, after a class, before a subtest and after a subtest. Setting any of these options will cause them to be in effect for all the selected tests.

Clicking this menu option opens the window shown in Figure 32.

Figure 32 **`cxtest` Global Test Parameters menu**



Command menu

The Command menu is used to perform actions on the node or complex being tested. These actions include:

- Go
- Reset Machine
- Read Boot Config Map

The Go selection starts the subtests. The subtests are sent to the test controller one at a time so that the application can detect the completion of each subtest. While running, an Abort button appears at the bottom of the screen. Clicking on this while the test is in progress terminates all test selections, not just the subtest currently running.

The Reset Machine option resets the system.

The Reading Boot Configuration Map must be used if the physical location of the boards being tested changes. It allows the user to keep `cxtest` running while the node is powered off and boards are moved.

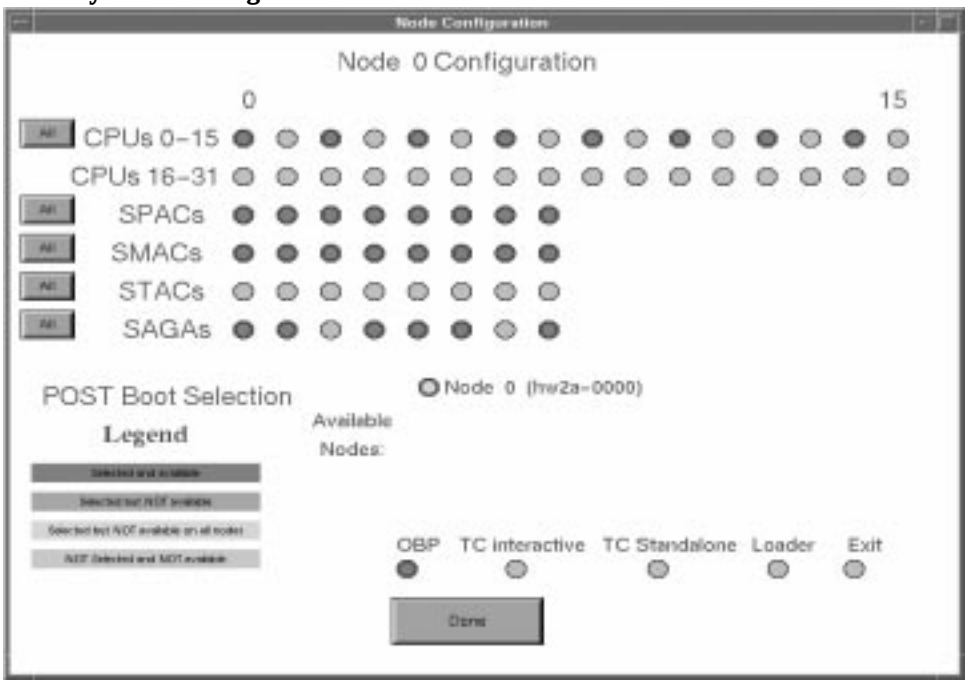
The Abort button will disappear while control is returned.

System Configuration menu

The System Configuration menu displays all nodes that were online at the time `cxtest` started. Clicking one of the menu entries opens a node configuration window (Node x Configuration window) that allows the user to select the hardware to test, excluding I/O-specific devices such as disk drives or PCI adapters. See Figure 33.

Any hardware selections made on this screen apply to all tests to be run. For example, to run test A with hardware configuration A and then test B with hardware configuration B, the configuration must be changed manually after test A is completed. The information contained in the node configuration window is extracted from the boot configuration map. If this map changes, it can be reread using the Command menu.

Figure 33 **System configuration window**



Help menu

The Help menu has two entries: About and Contents. The About selection displays the version number of `cxtest` running and the Contents selection opens a browser that can scroll through the help file.

Display area

The display area shows the output of the tests. This output consists of messages that indicate when the tests start, the amount of time that the test has be running, and any error information. The user can not cut and paste from this area. To record what transpires in test session, use the Log to File option.

When a test is started, a large ABORT button appears. Use this to escape out of a long test sequence or to stop a test not configured correctly. After using the ABORT button, reset the node from the Command menu. Once the abort command has been sent to the test controller, the button will disappear. The abort button also disappears when the test completes successfully.

Powering down the system

When using `cxtest` in a troubleshooting environment, it is not necessary to exit and enter `cxtest` each time the power is cycled.

To remove power to the system (for example, to move a board), power the system down leaving `cxtest` running. Make sure that no tests are actively running.

Once power is restored, POST returns control to the test controller in the stand-alone mode. The user must also wait for the `ccmd` routine to regenerate the database. A message will appear on the teststation console stating the database generation is complete.

After the database is regenerated, the Boot Configuration map must be read using the Command menu. If this is not done, `cxtest` will not have the correct hardware configuration information.

Command line interface

`cxtest` is a utility that allows the user to run tests loaded into the Test Controller. Tests can be specified on the command line or a Graphic User Interface can be started to simplify test selection. `cxtest` allows use of the Test Controller without being at the system console.

NOTE

The -d option must be used on the command line to start the GUI interface to `cxtest`.

By default, `cxtest` tries to load the test information needed from a file. The name of the file is cxtest.load located in the same directory as the test. A different file can be specified by using the `-f` option on the command line.

Command line options

When using the command line interface to `cxtest`, the command line should be built around the following example model.

`cxtest` command line model

cxtest [loading options] <test> [parameters]
[looping/pausing/control] [class/subtest selections] <test>[parameters]
[looping/pausing/control] [class/subtest
selections]...etc.

Table 23 Command line loading options.

Loading options	Description
`-f <load_file_name>`	Use the <load_file_name> to read what tests are to be loaded in to `cxtest`.
`default`	If no load options are specified, the file /spp/data/cxtest.load is read to know what tests to load.

Command line test selections

The command line interface deciphers the following switches to select tests.

- `-mem`—Memory diagnostic.
- `-io`—I/O diagnostic.
- `-cpu`—processor diagnostic.

All the arguments between two test selections apply only to first test specified as in the following example:

Example `cxtest` command line

```
cxtest -mem -lt 3 -c 4 -io -c 2
```

The looping specification only applies to the memory test which runs the class-4 tests three times. The I/O test run the class-2 test only once.

Command line looping and pausing

A number of different pausing options are available. Tests can be paused at the beginning of a subtest, end of a subtest, beginning of a class and at the end of a class. Table 24 shows `cxtest` looping and pausing options.

Table 24 Looping, pause, and control options

Looping and Pause Controls	Description
`-pe <ON-OFF>`	Pause at end of subtest
`-pb <ON-OFF>`	Pause at beginning of subtest
`-ps <ON-OFF>`	Pause at beginning of class
`-pt <ON-OFF>`	Pause at end of class
`-ls <number>`	Execute <number> of loops of the test that follows

Looping and Pause Controls	Description
`-lt <number>`	Execute <number> of loops of all the tests that follow
`-t <number>`	Switches the test controller to run on processor <number>. (range 0-15)
`-mt <number>`	Allows specification of error count

To set the number of times a test is looped on use the `-lt <number>` option.

Example of `cxtest -lt` option

```
cxtest -mem -lt 3 -c 4 -io -c 2
```

The looping specification only applies to the memory test which runs the class-4 tests three times. The I/O test run the class-2 test only once.

Command line error counts

The error count allows the test to proceed after an error has occurred. The error count must be set for the test to run after a failure. To set the error count use the `-mt <number>` option.

Command line class Selections

To select an entire class of subtests, use the `-c <number>` option. The user can specify a range of classes by using a hyphen between the numbers.

As an example, **`-c 2-4`**, runs classes 2,3 and 4.

To specify a list of classes, place a comma between the numbers. An example would be **`-c 5,7,2`**. This runs class 5 then class 7 and finally class 2.

Command line subtest selections

To select a subtest use the `-s <number>` option. To specify a range of subtests, use a hyphen between the numbers. As an example, **`-s 100-150`**, runs subtests 100 through 150.

To specify a list of subtests. place a comma between the numbers. As an example, **`-s 100,150,140`**, runs subtest 100, then subtest 150, and finally subtest 140.

Command line parameter specifications

To specify the value of a parameter for a test, use the `-pa# <val>` option. These options must be placed before the tests that uses them on the command line as in the following example:

Example of `cxtest -pa` option

```
cxtest -mem -c 1 -pa4 4 -pa5 2 -c 2
```

This runs class one, changes the value of the parameters 4 and 5, and then run class two. The parameters only have effect for the test specified. That is, if the memory test had been followed by `-io -c 1`, the value of parameters 4 and 5 would be the defaults for the I/O tests.

There are 128 parameters in all, `-pa0` through `-pa127`.

Changing test controller

The `-t <proc_num>` option changes the processor that is running the test controller. This parameter must be used before the test selections (i.e. `-c xx` or `-s yy`). There is a 10-second delay to invoke the change. `<proc_num>` must be a valid processor (0-31), and it must be present and available in the system.

Test output

Test progress and error information is displayed just as in the graphics interface, with the exception that the information is displayed on the terminal the test was started from. There is no logging to a file with this interface, so the invocation of `cxtest` should capture standard out and standard error into a file if the log is desired.

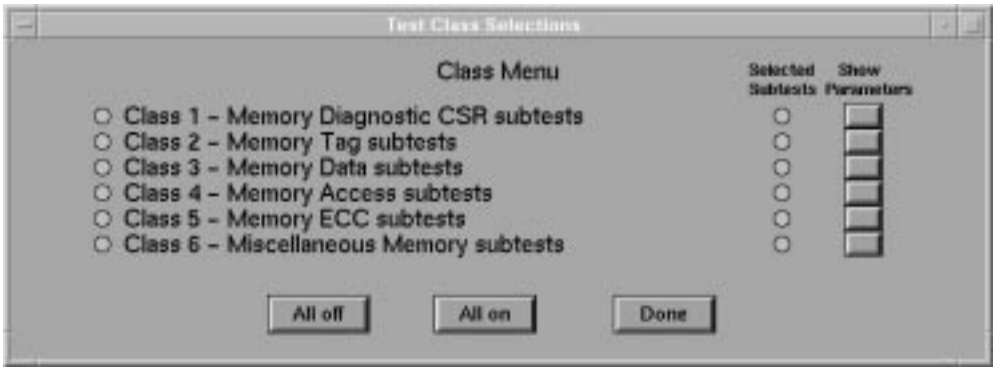
Example of running diagnostics from cxtest window

The following example procedure shows the user how to use mem3000 from cxtest. It assumes that the node configuration has been set up using the main cxtest window.

Step 1. From the cxtest main menu Tests option, select MEM3000 - EEPROM based memory tests.

This opens the class selection window shown in Figure 34.

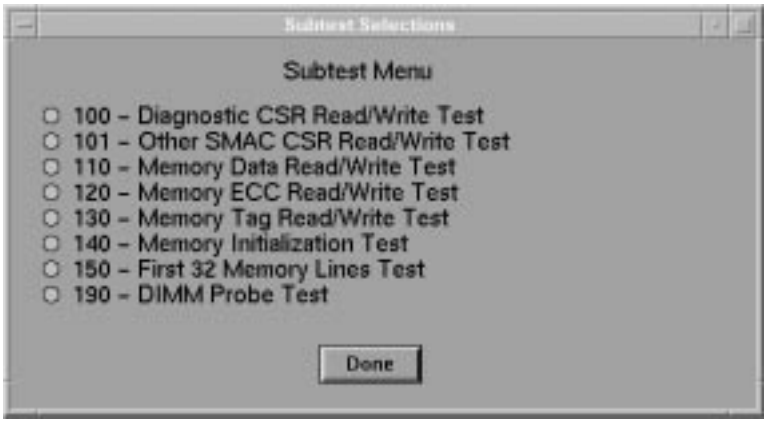
Figure 34 **mem3000 Test Class Selection window**



Step 2. In the Test Class Selections window, click on the round buttons to the left of the classes to select which class of test to execute. Any combination of classes may be selected.

Step 3. Click on a Selected Subtests button to select which associated subtests to run for each class. A Subtest Selections window opens for each Selected Subtests button clicked. The Class 1 Subtest Selections window is shown in Figure 35.

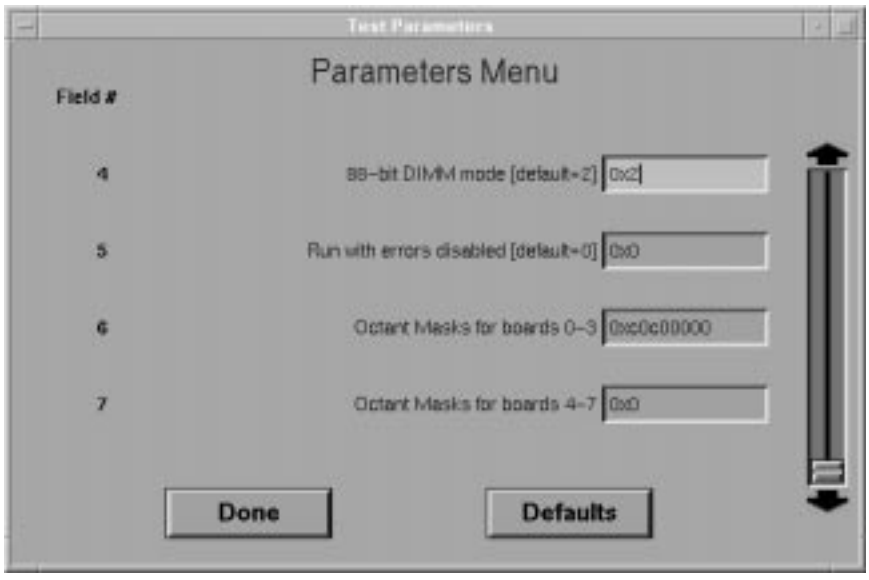
Figure 35 **mem3000 Class 1 Subtest Selections window**



Step 4. In the Subtest Selections window for each class, click the button for subtest to be executed. Any combination of subtests may be executed.

Step 5. To set the parameters for each class of test, click the appropriate Show Parameters button in Test Class Selections window. This opens the Class Parameters window. Figure 36 shows the mem3000 Class 1 Test Parameters window.

Figure 36 **mem3000 Test Parameters window**



Step 6. To start the selected tests and subtests, click the Go option in the Command menu in the `cxtest` main window.

Step 7. View the results in the lower window pane of the `cxtest` main window.

Example of running diagnostics from `cxtest` window

6 Processor-dependent code firmware loader

The processor-dependent code firmware loader (`pdcfl`) is a firmware module with the capabilities of loading other firmware modules into FLASH. It is intended to speed up download of POST and OBP on newly manufactured or malfunctioning utility boards. If the target system can successfully boot OBP, OBP should be used to download firmware in favor of `pdcfl`. Pdcfl can be loaded into FLASH using `load_eprom` as a stand-alone, potable module.

pdcfl loading, booting, and setup

NOTE This step should not be necessary under normal circumstances.

`pdcfl` is loaded on all Utility boards at the factory. If the utility board FLASH contents have been erased, pdcfl may be loaded into the Utility board using `load_eprom`. `load_eprom` supports a `-f` option for loading `pdcfl` to the appropriate sector in FLASH memory.

As an example:

```
load_eprom -n <node IP number|node IP name> -f /spp/firmware/pdcfl.fw
```

Once `pdcfl` has been loaded, it can be started by issuing a `do_reset` with a loader option:

Example of `do_reset` with loader option

```
do_reset <node id> loader
```

NVRAM setup

NOTE This step should not be necessary under normal circumstances.

If the NVRAM contents have been corrupted, there are two parameters that must be initialized: `ts_ip` and `scub_ip`. The usual values are:

```
ts_ip      15.99.111.99
scub_ip    15.99.111.116
```

`scub_ip` may vary based on the number of nodes connected to the test station. Use `ts_config` should to initialize the `scub_ip`. If `ts_ip` does not match the test station IP, use `pdcfl setenv`.

Teststation setup

When installing test station software, the install scripts automatically set up the test station to support `pdcfl`. If `pdcfl` is unable to access firmware files on the test station, correct the teststation configuration. The teststation needs to be setup to act as a `tftp` server for loading the desired files into FLASH memory.

This requires making these entries to the following files:

To /etc/services make the following entry:

```
tftp          69/udp          Trivial File Transfer Protocol
```

To /etc/inetd.conf make the following entry:

```
tftp        dgram  udp wait   root /usr/lbin/tftpd    tftpd -R 15
```

Also send a HUP to inetd.

To /etc/passwd make the following entry:

```
tftp:*:510:20::/spp/firmware:/usr/bin/false
```

Files for loading to FLASH can then be placed in the /spp/firmware directory.

pdcfl commands

From the `pdcfl` prompt, the following commands are supported:

- `printenv [variable]`—Prints configuration variables from NVRAM.
- `setenv variable value`—Allows setting configuration variables in NVRAM.
- `lifls`—Prints a listing of the LIF volume in the FLASH EEPROMs. The listing includes the name of the module, the FLASH address at which the module starts, the size in LIF units, the date the module was last written, and the sectors included by the module.

An example of the `lifls` command

```
PDCFL> lifls

LIF Volume FLASH4
Name         Addr           Size    Date           Sectors
-------------------------------------------------------------
POST         0xF0020000     0x400   04/09/97       4-5
TC           0xF0140000     0x300   04/09/97       16-17
CPU3000      0xF0170000     0x300   04/09/97       17-18
DIODC        0xF01A0000     0x300   04/09/97       19-20
MEM3000      0xF01D0000     0x2F0   04/09/97       20-21
RDR_DUMPE    0xF01FF000     0x10    04/09/97       21
IO3000       0xF0260000     0x500   04/09/97       25-27
INTER3000    0xF02B0000     0x300   04/09/97       27-28
PDCFL        0xF02E0000     0x200   04/09/97       29
DFDUTIL      0xF0300000     0x200   04/09/97       30
```

- `fload file location`—Loads a file from the teststation tftp directory to the address in FLASH specified in the LIF directory by name. `location` can also be a specific address given in hex to allow loading files that have not yet been entered in the LIF directory. If this form is used, the LIF directory will not be updated.

An example of the `fload` command

```
PDCFL> fload post.fw POST

TFTP server     : 15.99.103.191
CUB IP            : 15.99.111.150
Reading           : post.fw
Writing           : POST            (each '.' represents 4K copied)
Sector erased 0xF0020000
.....................................
Sector erased 0xF0040000
............
148384 bytes transferred
```

- reset [post]—Resets the node, optionally changing the boot vector to point to the POST module.

`pdcfl` does not currently support read/modify/write of sectors, so all sectors allocated in the LIF directory for a module are erased as the new file is written. If other modules exist in that sector, they are erased.

7 cpu3000

This chapter describes `cpu3000` processor test

`cpu3000` runs via the test controller and provides a basic test of the functionality of the PA8500. `cpu3000` requires a minimum of one processor with its associated SPAC and two EWMBs. Included in the testing are most of the instruction set, the ALU, general, space and control registers, external interrupts, RDRs, TLB RAM, the instruction cache, and the data cache. The tests are grouped together in five classes beginning with verification of the most basic functionality and progressing toward more complex functionality. Each class has a set of subtests that target specific functionality.

cpu3000 classes and subtests

cpu3000 consists of a series of tests grouped together in classes beginning with verification of the most basic functionality and progressing toward more complex functionality. Each class has subtests which target specific functionality.

When a failure is encountered, the chassis code is available through the test controller along with the progress value.

cpu3000 classes

cpu3000 has five classes of tests shown in Table 25.

Table 25 Classes of cpu3000 tests

Class	Name
1	Basic CPU tests
2	Instruction cache RAM test
3	Data cache RAM tests
4	TLB RAM tests
5	Functional tests requiring main memory

cpu3000 subtests

The cpu3000 subtests are listed in Table 26 through Table 29.

Table 26 **cpu3000 Class 1 subtests**

Subtest	Name	Description
100	Processor basic	Verifies the majority of registers and a basic set of instructions. Chassis code: 0x41020.
101	Processor-ALU	Verifies the processor and arithmetic Logic unit (ALU) functionality. Chassis code: 0x41021.
102	Processor branch	Verifies the branch instructions. Chassis code: 0x41022.
103	Processor-arithmetic condition	Verifies the arithmetic conditions of the unit, extract/deposit and carry/ borrow instructions.Chassis code: 0x41023
104	Processor bit operations	Verifies the processor's bit operation.Chassis code: 0x41024
105	Space and control registers	Verifies the space and control registers. Chassis code: 0x41025.
110	External interrupts	Executes sixty three external interrupts, one for each EIR_VAL position excluding Itimer. Chassis code: 0x41026.
111	Interval timer	Verifies the interval timer trap. its masking capability, associated control process, and timer rollover. Chassis code: 0x41027
120	Multimedia	Verifies the functional operation of the multimedia units. Chassis code: 0x41028.
130	Shadow	Verifies the shadow registers. Chassis code: 0x41029.

Subtest	Name	Description
140	Diagnostic register	Verifies the local Diagnose Registers. Chassis code: 0x4102a.
141	Remote diagnostics registers	Verifies the remote Diagnose Registers. Chassis code: 0x4102b.
150	Register bypass	Verifies the register bypass functionality of the processor. It tests three different types of bypassing that can occur between the two integer queues. Chassis code: 0x4102c.

Table 27 **cpu3000 Class 2 subtests**

Subtest	Name	Description
210	Icache RAM	This routine pattern tests the icache ram. Chassis code: 0x42020.

Table 28 **cpu3000 Class 3 subtests**

Subtest	Name	Description
310	Dcache RAM	Verifies the data cache rams. Chassis code: 0x42070.

Table 29 **cpu3000 Class 4 subtests**

Subtest	Name	Description
400	TLB RAM	Verifies the TLB ram arrays with a pseudo random pattern. Chassis code: 0x410b1.

Table 30 **cpu3000 Class 5 subtests**

Subtest	Name	Description
500	Late-early self test (LST-EST)	Runs subtests 100, 101, 102, 103, 104, 105, 120, 130, and 150, first in main memory and then in the Icache. This test has the following chassis codes: LST test - 0x44020 processor basic - 0x44021 processor ALU- 0x44022 processor branch - 0x44023 processor arithmetic condition - 0x44024 processor bit ops - 0x44025, space and control registers - 0x44026 multimedia - 0x44029 shadow - 0x4402a register bypass - 0x4402d.
510	Cache-byte	Verifies the instructions that store bytes, halfwords, and words.Chassis code: 0x44030.
520	Cache flush	Verifies the instructions that flush the Icache and Dcache. Chassis code: 0x44040.
530	Icache miss	Verifies that instructions can be encached from coherent memory. 0x44050.

Subtest	Name	Description
540	Dcache miss	Verifies that data can be encached from coherent memory. Chassis code: 0x44060.
560	TLB transfer	Verifies TLB hits and misses, as well as access rights and protection ID validation. Chassis code: 0x410b2.
570	Floating point unit	Verifies the floating point unit. It consists of several groups of tests that include testing of the FPU registers, instruction tests, trap handling, and access rights and ID validation. This test has the following chassis codes: FPU functionality - 0x410a0 FPU instruction - 0x410a2 FPU traps - 0x410a3 FPU miscellaneous tests- 0x410a4 FPU bypass - 0x410a5. FPU registers - 0x410a1

cpu3000 errors

When a failure occurs, the chassis code is available through the test controller, along with the progress value. The progress value indicates what portion of the subtest encountered the error.

cpu3000 errors

8 io3000

The I/O diagnostic supports Symbios 875 HVD SCSI controllers, Symbios 895 LVD SCSI controllers, and Tachyon Fibre Channel controllers.

`io3000` requires a node with a minimum of one processor, one SIOB with associated SPACs, and two EWMBs with associated SMACs. To exercise peripherals, either a Symbios SCSI or a Tachyon Fibre Channel card is required.

io3000 classes and subtests

`io3000` consists of a series of tests grouped together in classes beginning with verification of the most basic functionality and progressing toward more complex functionality. Each class is broken down into subtests which target specific functionality.

The following sections describe the classes and individual subtests.

io3000 classes

`io3000` has 10 classes of tests shown in Table 31.

Table 31 Classes of `io3000` tests

Class	Name	Description
1	SAGA CSR Test	Verifies successful writes and reads of SAGA CSRs.
2	SAGA Memory Test	Verifies the functionality of SAGA context/shared memory and prefetch memory.
5	SCSI Disk Interface Test	Verifies the ability to successfully issue SCSI commands to every selected disk or Fibre Channel target.
6	Channel Mode Test	Verifies the ability to successfully build and use SAGA channels in all the supported modes. Also, every channel can be verified to be usable.
7	DMA Boundary Conditions Test	Verifies that various DMA conditions and every possible interrupt vector work correctly. Also verifies every possible interrupt vector.
8	MultiDisk Concurrency Test	Queues up all selected disks for simultaneous transfers. In this test, all disks operate in a parallel fashion.

Class	Name	Description
11	SAGA SCSI Tape Interface Test	Verifies the ability to successfully issue SCSI commands to every selected tape drive.
12	Symbios Test	Verifies the basic functionality of the Symbios SCSI controller.
15	CDROM SCSI Access Test	Verifies basic SCSI bus access.
16	Tachyon SAGA PCI Access Test	Verifies the SAGA PCI interface to all selected Tachyon controllers

io3000 subtests

The `io3000` subtests are listed in Table 32 through Table 41.

Table 32 **`io3000` Class 1 subtests**

Subtest	Name	Description
100	CSR reset	Verifies that each SAGA CSR has a defined state and contains the proper value after the SAGA reset is completed.
105	CSR read/ write	Verify writes and reads for each SAGA CSR using a bitwise March C- test.
110	Error CSR	Verifies that each individual error type in the ErrorCause register can be set and is capable of generating an interrupt.

Table 33 **io3000 Class 2 subtests**

Subtest	Name	Description
200	Context/ shared memory read/ write	Writes to the first 64-bit location of each context SRAM and reads them to verify that they can be uniquely accessed.
205	Context/ shared memory access width	Verifies that all supported access widths of context SRAM function properly by writing and reading the first 64-bit location.
210	Context/ shared memory march C-	Verifies that coverage for full march C- will increase from approximately 99% to 100% of targeted fault using a bitwise march C- algorithm.
215	Context/ shared memory pattern	Writes and reads random pattern to all of context/shared memory. The random pattern can be modified by changing the random seed option. Also, a user-specified pattern can be used by setting the user pattern options.
220	Context/ shared memory parity detection	Verifies the ability of SAGA to detect parity errors on reads from context/ shared memory. This test uses the FCC bit (force parity error to context SRAM) to write a parity error into context SRAM. The bad parity is read out, causing a parity error to be detected and logged in the SAGAs Error Cause CSR.
225	Prefetch memory read/ write	Verifies that the first 64-bit location of each prefetch SRAM uniquely accessed.
230	Prefetch memory access width	Verifies that all supported access widths of prefetch SRAM function properly by writing and reading the first 64-bit location.

Subtest	Name	Description
235	Prefetch memory march C-	Verify writes and reads to all of prefetch memory using a bitwise march C- algorithm. The default option does a shortened version of the march C- algorithm by using a limited pattern set. The march C- complete enable can be set to do a full march C- test. The test time increases by a factor of approximately four. The fault coverage for full march C- increases from approximately 99% to 100% of targeted faults.
240	Prefetch memory pattern	Writes and reads random pattern to all of prefetch memory. The random pattern can be modified by changing the random seed option. Also, a user-specified pattern can be used by setting the user pattern options.
245	Prefetch memory parity detection	Verifies the ability of SAGA to detect parity errors on reads from prefetch memory. This test uses the FPR bit (force parity error to prefetch SRAM) to write a parity error into context SRAM. Then the bad parity is read out, causing a parity error to be detected and logged in the SAGA's Error Cause CSR.

Table 34 **`io3000` Class 5 subtests**

Subtest	Name	Description
500	SCSI disk test unit ready	A SCSI `test unit ready` command is issued to all selected devices at least twice. This first time, it should return with a SCSI check condition (not reported to the user) since the SCSI bus has been reset. The command is retried after approximately one second. If the second test unit ready fails, an error is reported. The `test unit ready` command does not cause a SCSI data phase to occur.
505	SCSI disk inquiry	A SCSI `inquiry` command is executed on every selected device. This test verifies that the device type field in the inquiry return data is a direct access (disk). A SCSI data in phase will occur.
510	SCSI disk read capacity	A SCSI `read capacity` command is issued to every selected device.
515	SCSI disk read	A SCSI `read` command is issued to every selected device. No data verification is performed.
520	SCSI disk write	A SCSI `write` command is issued to every selected device. No data verification is performed. This test only writes to the disk if the write enable option is turned on. The default is to not allow writes to the device.

Table 35 **io3000 Class 6 subtests**

Subtest	Name	Description
600	Channel init, ATPR = 0x0	
625	Channel init, write tlb, data prefetch, ATPR = 0xa	
630	Channel init, tlb prefetch, ATPR = 0xc	
635	Channel build, ATPR = 0xc	
640	Channel init, tlb & data prefetch, ATPR = 0xe	
645	Channel build, ATPR = 0xe	
650	Channel context access	Verifies selected SAGA channels in virtual mode. After each channel is built, the test checks the context SRAM. If the full channel disable option is set, Channels 0, 1007, and power of 2 channels greater than 31 are tested. Otherwise all channels greater than channel number 31 are tested. Channels 1-31 are reserved for controller DMA access. (default).
605	Channel build, ATPR = 0x0	

Subtest	Name	Description
610	Channel init, data prefetch, ATPR = 0x2	
615	Channel build, ATPR = 0x2	
620	Channel init, write tlb, ATPR = 0x8	

Subtests 600-645 create channels by writing to the SAGA channel builder CSR. The method of channel creation and the specific mode (ATPR setting) is specified in the subtest's one line description. Each test will write data to the disk and read it back and verify it. Each disk's write enable option must be set for the writes and data verification to be allowed.

Table 36 **`io3000` Class 7 subtests**

Subtest	Name	Description
700	External interrupt	Verifies all possible external interrupt vectors. A separate DMA is executed for each external interrupt vector.
705	DMA across page and channel	Verifies writes and reads of DMAs that cross page and channel boundaries.
710	Jump forward within a page	Verifies writes and reads of DMAs that jump forward within a page.
715	Jump backward within a page	Verifies writes and reads of DMAs that jump backward within a page.
720	Jump outside of a page (TLB encached)	Verifies a DMA jump outside of a page. The TLB for the destination page is encached in context SRAM for both writes and reads.

Subtest	Name	Description
725	Jump outside of a page (TLB not encached)	Verifies a DMA jump outside of a page. The TLB for the destination page is not encached in context SRAM. This means that SAGA must fetch a new TLB before the transfer can continue. This is done for both writes and reads.
730	Jump outside of a channel	Verifies a DMA jump outside of the current channel. This is done for both writes and reads.
735	Non contiguous TLBs	Sets up a translation table for scattered system page mappings (noncontiguous). Then a DMA is set up to use this table. This causes the SAGA to access pages noncontiguously throughout the DMA.

Table 37 **io3000 Class 8 subtests**

Subtest	Name	Description
800	Multidisk nonmixed traffic	Issues all selected devices simultaneous SCSI writes and then SCSI reads. The channels are programmed in virtual mode, with data and TLB prefetch turned on.
805	Multidisk mixed traffic, ATPR = 0xe	All selected devices transfer data simultaneously. Some devices are performing SCSI reads, while others are performing SCSI writes, thereby causing mixed or bidirectional traffic on the SCSI and PCI busses. The channels are programmed in virtual mode, with data and TLB prefetch turned on. Refetch is turned off.
810	Multidisk mixed traffic, ATPR = 0xf	This is the same subtest as 805 above, but with refetch turned on.

Table 38 **io3000 Class 11 subtests**

Subtest	Name	Description
1100	SCSI tape test unit ready	Issues a SCSI test unit ready command to all selected devices at least three times. This first time the SCSI bus will have been reset. This is normal. The command is retried after approximately one second. The command is issued again to allow for a check condition due to the medium being changed. Many tape drives require a tape to be installed in the drive, causing the second test unit ready to respond with a medium changed sense status. If the third test unit ready fails, an error is reported. The test unit ready command does not cause a SCSI data phase error to occur.
1105	SCSI tape inquiry	Executes a SCSI inquiry command on every selected device. It verifies the device type field in the inquiry return data to be sequential (tape). A SCSI data in phase error does occur.
1110	SCSI tape rewind	Executes a SCSI rewind command on every selected device and waits for it to complete. The rewind command will not cause a SCSI data phase to occur.
1115	SCSI tape read	If only fixed block sizes are supported or 255 bytes are supported, this test executes a SCSI read command after one block. A SCSI data in phase does occur; no data verification is performed.

Table 39 **io3000 Class 12 subtests**

Subtest	Name	Description
1200	Symbios PCI configuration space test	Verifies the ability of the SAGA to access the Symbios SCSI controller by way of the PCI configuration space. Verifies the PCI vendor ID and device ID fields to be 0x1000 and 0x000f, respectively. Also verifies the base address registers to be writable and readable.
1205	Symbios SCSI PCI I/O and Memory space test	Maps the Symbios SCSI controller through PCI configuration space so that the controller's CSRs may be accessed by way of both PCI I/O and memory space. The test writes a pattern to a scratch register (SCRATCHA) in the Symbios chip. The register is then read back to verify the previous write succeeded.

Subtest	Name	Description
1230	Symbios SCSI Scripts RAM test	Performs a simple `data equals address` pattern test of the SCRIPT RAM.
1240	Symbios SCSI Interrupt test	Copies a simple `SCRIPTS` instruction to SCRIPTS RAM on the Symbios controller. The SCRIPTS instruction is a simple `INT` opcode which, when executed by the Symbios chip, should cause a DMA interrupt to be logged. The DSP register of the Symbios chip is set to point to the instruction and the ISTAT register is polled until the interrupt is detected or the allotted time has elapsed.
1250	Symbios SCSI DMA engine test	Writes a simple `SCRIPT` to Symbios SCRIPTS RAM which contains a `MEM MOVE` opcode. `SCRIPT` copies 256 bytes from one section of SCRIPTS RAM to another SCRIPTS RAM area. Once the SCRIPT has completed, the test verifies that the original block of data was copied to the destination area.

Table 40 io3000 Class 15 subtests

Subtest	Name	Description
1500	SCSI CDROM test unit ready	Issues a SCSI `test unit ready` command to all selected devices at least twice. The response to first command should return a SCSI "check condition" (not reported to the user) since the SCSI bus will have been reset. After approximately one second, the command is sent again. If the second test unit ready fails, an error is reported. The test unit ready command will not cause a SCSI data phase to occur.
1505	SCSI CDROM inquiry test	Executes a SCSI `inquiry` command on every selected device. Verifies the device type field in the inquiry return data to indicate a CDROM.
1510	SCSI CDROM read capacity test	Issues a SCSI `read capacity` command to every selected device.
1515	SCSI CDROM read test	Issues a SCSI `read` command to every selected device. No data verification is performed.

NOTE Class 15 subtests will also test DVD drives.

Table 41 **io3000 Class 16 subtests**

Subtest	Name	Description
1600	Tachyon PCI configuration space test	Verifies the ability of the SAGA to access the Tachyon Fibre Channel controller by way of the PCI configuration space. Verifies the PCI vendor ID and device ID fields to be 0x107e and 0x0004, respectively. Also verifies the base address registers to be writable and readable.
1605	Tachyon PCI I/O and Memory space test	Maps the Tachyon Fibre Channel controller through PCI configuration space so that the controller's CSRs may be accessed by way of PC memory space. The test writes a pattern to the world-wide name Hi (www_hi) in the Tachyon chip. The register is then read back to verify the previous write succeeded.

User parameters

The test controller provides `io3000` with up to 37 user parameter words. Current parameters are defined in Table 42.

Table 42 **io3000 test parameters**

Words	Description
0	See Table 43.
1	Device write enable mask—Each bit in the mask corresponds with a device. Bit 0 (MSB or left most bit in the parameter word) corresponds to device 0, bit 29 corresponds to the last (29th) device. Device 0 is the first device parameter location in user parameter word 8 (see Words 8-19 Device specification below). A binary '0' in a device's bit field means that SCSI writes (to that disk) are not enabled. Any test that does SCSI writes will not do so when the disk's corresponding write enable is turned off (binary '0'). The subtest will not be completely disabled though. This means that SCSI reads will still take place, but data verification will not be performed. The default setting for all disks is SCSI writes are disabled.
2	Transfer length (class 8 only)
3	Pattern (upper 32 bits)
4	Pattern (lower 32 bits)
5	Random seed
6	Custom SCSI firmware file length—If a custom (nonsupported) SCSI controller firmware file is used, the length in halfwords must be entered in this parameter. This parameter is ignored if the Custom SCSI firmware enable is not set.
7	Not used.
8-37	Device specification. See Figure 37 on page 150.

Table 43 **io3000 user test parameter word 0 bit definition**

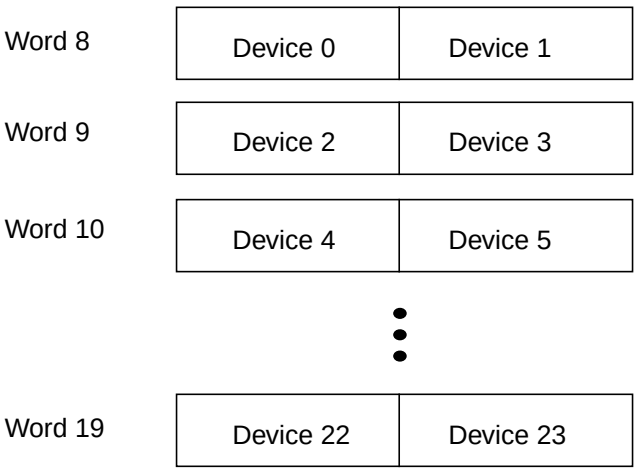
Bit	Description
0-23	Unused
24	Force code copy enable—Setting this bit causes all subtests that use encached routines to copy the code segment from flash into main memory. The copy will be performed even if the previous subtest already performed the copy. This feature should not be needed unless the code in main memory is being corrupted in a manner that cannot be easily detected.
26	Full channel test disable (subtest 650)— Setting this bit causes subtest 650 to only test channels that are a power of 2, with the addition of channel 1007 (default). This reduces the run time significantly on this subtest.
28	Multidisk enable (classes 6-7)—Setting this bit causes all specified disks to be tested in classes 6-7. The default is to only test the first disk (as specified in the user parameters) on each controller. Since classes 6-7 do disk transfers serially, little additional coverage is gained by running the tests on all the disks.
29	User pattern enable (subtests 215, 240, classes 6-8)—Setting this bit causes each of the above specified tests to use the patterns as specified in user parameter 3 and 4, rather than a hard coded default pattern.
30	Random pattern enable (subtests 215, 240, classes 6-8)—Setting this bit causes each of the above specified tests to use subtest specific random patterns, rather than a hard coded default pattern.
31	March C complete (subtests 210, 235)—Setting this bit causes each of the above specified tests to do a complete bitwise march C test on the SRAM. The default is to do a quick version which takes about 25 percent of the time with about 99 percent of the coverage.

Device specification

Due to Core Logic SRAM space limitations, only 20 devices per SAGA can be tested at a time. Up to 24 SCSI devices can be specified using parameter words 8-19. Each of these parameter words contains two device specifications, as shown in Figure 37. Word 8 contains device specification 0 and 1. Word 9 contains 2 and 3, and so on.

Up to six Fibre Channel devices can be specified in parameter words 20-37. Each device requires three parameter words as shown in Figure 38 and Table 45.

Figure 37 **`io3000` test parameter device specification for directly attached SCSI targets (words 8-19)**

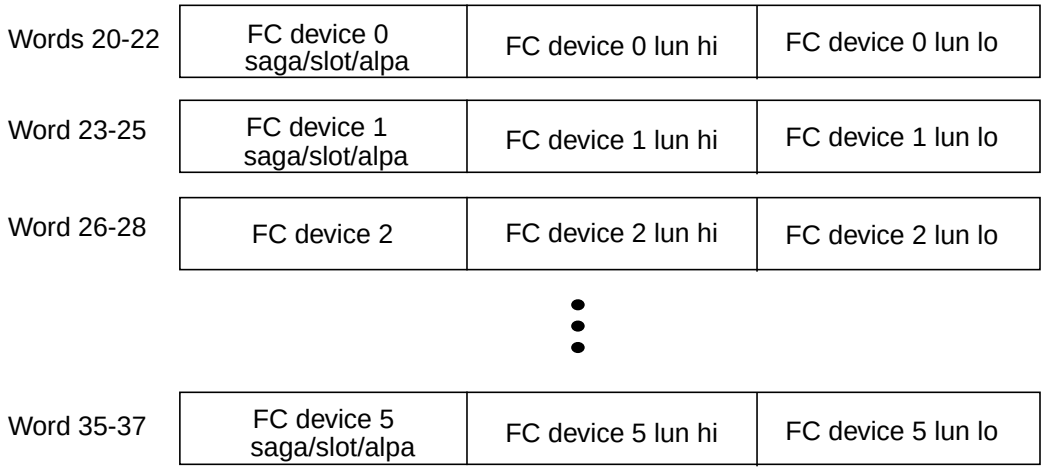


Fields within each parameter word specify the devices as shown in Table 44. Bit 0 is the upper (left most) bit in the parameter word.

Table 44 **io3000 bit definition for direct SCSI device specification (words 8-19)**

Bit	Definition
0-3	SAGA
4-7	Slot
8-11	SCSI target
12-15	SCSI lun
16-19	SAGA
20-23	Slot
24-27	SCSI target
28-31	SCSI lun

Figure 38 **io3000 test parameter device specification for Fibre Channel attached SCSI targets (words 20-37)**



Fields within each parameter word specify the devices as shown in Table 45. Bit 0 is the upper (left most) bit in the parameter word.

Table 45 **`io3000` bit definition for Fibre Channel attached SCSI device specification (words 29-37)**

Location	Bit	Definition
Word n	0-3	SAGA
Word n	4-7	Slot
Word n	8-31	AL_PA (or D_ID)
Word n+1	0-31	FC lun hi
Word n+2	0-31	FC lun lo

Devices are numbered according to their position in the parameter list. A device can be specified in any of the device specification locations in user parameter space. An unused device parameter should be initialized such that the slot field is 0xf (that is, device specification of 0x0f00). Therefore, if both device parameters in a given parameter word are unused, the parameter word would be set to 0x0f000f00.

As an example, to specify a disk on SAGA 0x4, slot 0x2, SCSI identification 0xa, SCSI lun 0x0, set parameter word 8 to 0x42a00f00. The lower (right) half of the parameter word has the slot field set to the 0xf. The device number is 0 since it was entered in device 0 parameter location.

When using cxtest to run `io3000`, SAGAs are referred to according to their textual name as stamped in the node sheet metal. Table 46 correlates SAGA names with SAGA numbers:

Table 46 **io3000 SAGA name to number correlation**

SAGA name	SAGA number
IOLF_A	4
IOLF_B	0
IOLR_A	5
IOLR_B	1
IORR_A	6
IORR_B	2
IORF_A	7
IORF_B	3

io3000 error codes

When a failure is encountered, an event code is set along with an error message. The least significant 12 bits of the event code contain the error code. Table 47 lists the `io3000` error codes.

io3000 general errors

`io3000` general error codes post no error messages. Table 47 shows each `io3000` general error code.

Table 47 `io3000` general error codes

Code	Description
0x1	Core logic SRAM allocation failure. This is a software error that indicates that the software has run out of core logic SRAM to store internal data structures.
0x2	Interrupt allocation failure. This is a software error that indicates that the software has run out of available external interrupt vectors.
0x3	No device specified. `io3000` was looking for a device in the user parameters and found none.
0x4	An invalid combination of processors has been selected. Due to the shortage of core logic SRAM, the per processor stack space is only 1Kbytes. This has proven to be inadequate for portions of `io3000`. Therefore, processor selection has been limited such that adjacent processors cannot be selected simultaneously. Also, processor 0xf can not be used.
0x5	A random number seed of 0 was specified. The seed must be nonzero.

io3000 device specification errors

io3000 device specification errors post the following error message:

SAGA_name/ctlr_num/tgt_num/lun_num

Example of io3000 device specification error message:

IOLF_A/ct0/idf/lu0

Table 48 shows each io3000 general error code.

Table 48 io3000 device specification error codes

Code	Description
0x8	Duplicate device specification. The same device was specified multiple times in the user parameters.
0x9	Invalid SAGA number. The number in the SAGA field of one of the device parameters is invalid (> 7).
0xa	Invalid slot number. The number in the slot field of one of the device parameters is invalid.
0xb	Invalid logical unit number. The number in the logical unit field of one of the device parameters is invalid (> 7).
0xc	Duplicate Fibre Channel device specification.
0xd	Invalid Fibre Channel SAGA number.
0xe	Invalid Fibre Channel slot number.
0xf	Invalid Fibre Channel LUN number.

io3000 SAGA general errors

io3000 SAGA general errors post the following error message:

SAGA_name

Example of io3000 SAGA general error message:

IOLF_B

Table 49 shows each io3000 SAGA general error code.

Table 49 io3000 SAGA general errors

Code	Description
0x10	An SAGA specified in the user parameters was not available.
0x11	Unable to reset SAGA. io3000 was unsuccessful in setting or resetting the SAGA online bit on it's associated SPAC.
0x12	Data prefetch timeout. The prefetch valid bits in the channel context never became valid, or did so too slowly.

io3000 SAGA CSR errors

io3000 SAGA CSR error codes post the following error message:

SAGA_name/address/act_val/exp_val

Example of io3000 SAGA CSR error message:

```
IOLF_B/fc010008/00e0000f0c000000/00e0000f0c100000
```

Table 50 shows each io3000 SAGA CSR error code.

Table 50 io3000 SAGA CSR errors

Code	Description
0x20	SAGA CSR failure.
0x21	SAGA PTE failure
0x22	SAGA read TLB even failure.
0x23	SAGA read TLB even failure.
0x24	SAGA read TLB odd failure.
0x25	SAGA write TLB even failure.
0x26	SAGA write TLB odd failure.

io3000 SAGA ErrorInfo CSR error

The io3000 ErrorInfo CSR error code posts the following error message:

SAGA_name/cause_bit/address/act_val

Example of io3000 SAGA ErrorInfo CSR error:

IOLF_A/5/fc210098/10e0000f0c000000

Table 51 shows the io3000 SAGA ErrorInfo CSR error code.

Table 51 io3000 SAGA ErrorInfo CSR error

Code	Description
0x50	SAGA ErrorInfo CSR failure.

io3000 SAGA ErrorCause CSR errors

io3000 SAGA ErrorCause CSR error codes 0x54 and 0x55 post the following error message:

SAGA_name/address/act_val

Example of io3000 SAGA ErrorCause CSR error message for 0x54 and 0x55 codes:

IOLF_A/fc210080/0000010000000000

io3000 SAGA ErrorCause CSR error code 0x58 posts the following error message:

SAGA_name/ctlr_num/address/act_valPIC_name/address/act_val

Example of io3000 SAGA ErrorCause CSR error message for 0x58 code:

IOLF_A/ct1/fc210108/0010000000000000/fc210080/0000010000000000

Error 0x58 occurs when a bit in the controller's corresponding SAGA PCIxStatCSR is set. Specifically, the bits that cause this error are SawAddrParErr, BrokenDev, and SawDataPtyErr.

Table 52 shows each io3000 SAGA ErrorCause CSR error code.

Table 52 `io3000` SAGA ErrorCause CSR errors

Code	Description
0x54	SAGA ErrorCause CSR failure.
0x55	SRAM parity error expected. This error occurs when the cci_rdperr bit in the SAGA ErrorCause does not get set when SRAM parity errors are forced.
0x58	PCIx status failure.

io3000 SAGA SRAM errors

`io3000` SAGA SRAM error codes post the following error message:

SAGA_name/address/act_val/exp_val

Example of `io3000` SAGA SRAM error message:

```
IOLF_A/f81fc00080/5555555555555555/55f5555555555555
```

Table 53 shows each `io3000` SAGA SRAM error code.

Table 53 `io3000` SAGA SRAM errors

Code	Description
0x60	SRAM access failure. `io3000` was unable to successfully write and read SRAM on the SIOB.
0x61	SRAM march C failures. A failure was detected during the march C test of SRAM on the SIOB. The range of codes refer to incremental stages of the march C algorithm as follows: none - write ~patt (->) 0x61 - read ~patt, write patt (->) 0x62 - read patt, write ~patt (->) 0x63 - read ~patt, write patt (<-) 0x64 - read patt, write ~patt (<-) 0x65 - read ~patt (<-)
0x66	SRAM read access width test failed.
0x67	SRAM write access width test failed.
0x68	SRAM pattern test failure.

io3000 controller general errors

io3000 Controller general error codes post the following error message:

SAGA_name/ctlr_num

Example of io3000 controller general error message:

```
IOLF_B/ct0
```

Table 54 shows each io3000 general controller error code.

Table 54 io3000 Controller general errors

Code	Description
0x80	The controller was not detected as present per the SAGA's PcixStatCSR PCI card present bits.
0x81	SCSI flash read error. io3000 was unable to successfully read the SCSI controller's flash memory.
0x82	io3000 was unable to initialize the controller.
0x83	The loopback test on the controller failed.
0x84	The controller was unexpectedly offline.

io3000 PCI errors

io3000 PCI error codes post the following error message:

SAGA_name/ctlr_num/address/act_val/exp_val

Example of io3000 PCI error message:

```
IOLF_B/ct1/f804000010/ffffff01/00000001
```

Table 55 shows each io3000 PCI error code.

Table 55 `io3000` PCI errors

Code	Description
0x90	PCI vendor id failure. `io3000` was unable to successfully read the controller's PCI vendor id
0x91	PCI device id failure. `io3000` was unable to successfully read the controller's PCI device id.
0x92	PCI io base address register failure. `io3000` was unable to successfully read and write the controller's PCI io base address register.
0x93	PCI memory base address register failure. `io3000` was unable to successfully read and write the controller's PCI memory base address register.
0x9A	Symbios SCRATCHA register failure. `io3000` was unable to successfully read and write the controller's SCRATCHA register.

io3000 controller command errors

`io3000` controller command error codes post the following error message:

```
SAGA_name/ctlr_num/tgt_num/lun_num/comp_stat/
scsi_stat:sense_key:sense_code:sense_code_qualifier
```

Example of `io3000` controller command error message:

```
IOLF_A/ct0/idf/lu0/comp:0/scsi:2
```

Table 56 shows each `io3000` controller command error code.

Table 56 `io3000` controller command errors

Code	Description
0xc0	SAGA command completion failure. This means a queued command has failed and has a nonzero completion status.
0xc1	SCSI status failure. This means a SCSI command has terminated with nonzero SCSI status.

io3000 DMA error

The io3000 DMA error code posts the following error message:

SAGA_name/ctlr_num/tgt_num/lun_num/address/act_val/exp_val

Example of io3000 DMA error message:

IOLF_A/ct0/idf/lu0/0004148200/a5a5a5a4/a5a5a5a5

Table 57 shows the io3000 DMA error code.

Table 57 io3000 DMA error

Field	Description
0xd0	Data miscompare on DMA. Data in the destination buffer does not match data in the source buffer.

io3000 SCSI inquiry error

The io3000 SCSI inquiry error code posts the following error message:

SAGA_name/ctlr_num/tgt_num/lun_num/act_val/exp_val

Example of io3000 SCSI inquiry error message:

IOLF_A/ct0/idf/lu0/1/0

Table 58 shows the io3000 SCSI inquiry error code.

Table 58 io3000 SCSI inquiry error

Code	Description
0xe0	Wrong peripheral device type found in SCSI inquiry return data.

io3000 Symbios controller specific errors

io3000 Symbios controller specific error codes post the following error message:

SAGA_name/ctlr_num/address/act_val/exp_val

Example of `io3000` Symbios controller specific error message:

```
IOLF_B/ct1/f804000010/ffffff01/00000001
```

Table 59 shows each io3000 Symbios controller specific error code.

Table 59 `io3000` Symbios controller specific errors

Code	Description
0x110	General failure detected on Symbios controller.
0x113	Error detected during SCRIPTS RAM pattern testing.
0x114	Interrupt test failed. The address is the address of the interrupt register. The expected data contains the bit of that interrupt register expected to be set, while the actual data contains the entire contents of the ISTAT or DSTAT register.
0x115	Symbios DMA test failed.

io3000 Tachyon controller specific errors

`io3000` Tachyon controller specific error codes post the following error message:

```
SAGA_name/ctlr_num/address/act_val/exp_val
```

Example of `io3000` Tachyon controller specific error message:

```
IOLF_B/ct1/f804000010/ffffff01/00000001
```

Table 59 shows each io3000 Symbios controller specific error code.

Table 60 `io3000` Symbios controller specific errors

Code	Description
0x90	PCI vendor ID not as expected.
0x91	PCI device ID not as expected.
0x93	PCI memory address Base Register write/read fail.

io3000 DIODC driver errors

`io3000` Diagnostic I/O Dependent Code (DIODC) driver error codes post the following error message:

SAGA_name/ctlr_num/tgt_num/lun_num/ctlr_status/dev_status

Example of `io3000` DIODC driver error message:

```
IOLF_A/ct1/ct0/idf/lu0/81/0
```

Table 61 shows each `io3000` Symbios controller specific error code.

Table 61 `io3000` DIODC controller specific errors

Code	Description
0x120	General controller error.
0x121	No controller detected in the selected slot.
0x122	Unsupportable controller detected.
0x130	General failure detected.
0x131	Attempted to open a device. An open consists of a SCSI Test Unit Ready followed by a SCSI Inquiry command. See the controller status codes for more details.

Table 62 Symbios controller status codes

Code	Description
0x81	Symbios Queue Overflow.
0x82	Symbios Queue Empty.
0x88	Invalid handle.
0x84	Timeout during select.
0x85	Timeout detected waiting for a SCRIPT to complete.
0x86	Device transitioned to an unexpected phase.
0x87	Device in an undefined SCSI phase.
0x88	Target not online.

Notes on io3000

`io3000` dumps trace data into Core Logic SRAM to troubleshooting failures. A script provided with `io3000` called `io_tr` is located in the scripts directory (located in /spp/scripts at the time of this writing) that views this trace data. `io_tr` prints the version of `io3000` from which it was built. If the versions does not match, there is no guarantee that the information presented will be correct.

9 mem3000

This chapter describes `mem3000`, a memory test for V2500 systems.

`mem3000` is core logic flash-based memory diagnostic that verifies the functionality of the memory subsystem.

`mem3000` requires a node with a minimum of one processor with two memory boards. Excalibur W Memory Boards (EWMBs) must be installed in pairs in order for the test to properly execute.

mem3000 classes and subtests

`mem3000` verifies the V2500 memory subsystem using the Test Controller.

`mem3000` requires one node with a minimum of one process with associated SPAC and two EWMBs with associated SMACs.

`mem3000` consists of a series of tests grouped together in classes beginning with verification of the most basic functionality and progressing toward more complex functionality. Each class has several subtests that target specific functionality.

mem3000 classes

`mem3000` has six classes of tests shown in Table 63.

Table 63 **`mem3000` test classes**

Class	Description
1	Verifies the operation of the diagnostic CSRs on each EMB.
2	Verifies the tag field.
3	Verifies the data field.
4	Verifies the various coherent and noncoherent transactions.
5	Verifies the ECC.
6	Verifies miscellaneous memory capabilities.

Class 1 and class 2 subtests (with the exception of subtest 150) can be configured to test a single EMB. Subtest 640 can also be used to test a single EMB.

Running any other Class 4, 5, or 6 subtest with only one EMB selected is not recommended.

Class 3 subtests and subtest 150 use memory interleaving and do not work with a single EMB selected.

mem3000 subtests

The mem3000 subtests are listed in Table 64 through Table 69.

Table 64 **mem3000 class 1 subtests**

Subtest	Description
100	Verifies the diagnostic CSRs can be written and read
101	Verifies the other SMAC CSRs can be written and read
110	Verifies data can be written and read on each DIMM using the diag CSRs
120	Verifies ECC can be written and read on each DIMM using the diag CSR
130	Verifies the tag can be written and read on each DIMM using the diag CSRs
140	Verifies memory lines on each DIMM can be initialized using the diag CSRs
150	Verifies the first 64 memory lines of each EWMB using various data patterns
190	Verifies that each DIMM passes DIMM probing similar to the POST DIMM probe.

Table 65 **mem3000 class 2 subtests**

Subtest	Description
200	Verifies the tag portion of a memory line using different patterns
210	Verifies the tag portion of a memory line using an addressing pattern
211	Verifies the tag portion of a memory line using a byte uniqueness pattern, i.e. 0x0001020304050607
230-238	Verifies the tag portion of a memory line using the MarchC algorithm and different patterns

Table 66 **mem3000 class 3 subtests**

Subtest	Description
300	Verifies the memory lines on each DIMM can be written and read using coherent operations
310	Verifies the data portion of a memory line using an addressing pattern with coherent operations
311	Verifies the data portion of a memory line using a byte uniqueness pattern with coherent operations
330-338	Verifies the data portion of a memory line using the MarchC algorithm and different patterns with coherent operations

Table 67 **mem3000 class 4 subtests**

Subtest	Description
400	Verifies load and store transactions to memory
410	Verifies data flush transactions to memory
420	Verifies non-coherent transactions to memory

Table 68 **mem3000 class 5 subtests**

Subtest	Description
500	Verifies ECC single bit data portion errors are detected, logged, and corrected
501	Verifies ECC single bit tag portion errors are detected, logged, and corrected
502	Verifies ECC single bit ECC portion errors are detected, logged, and corrected

Subtest	Description
510	Verifies ECC double bit data errors are detected and logged using coherent operations
520	Verifies ECC double bit data errors are detected and logged using non-coherent operations
530	Verifies that ECC errors are ignored when disabled

Table 69 **mem3000 class 6 subtests**

Subtest	Description
600	Verifies the memory system detects and reports accesses to all illegal and/or invalid memory space
610	Verifies the memory system detects and reports error conditions when the memory tag state is ERROR
640	Determines whether 80-bit or 88-bit DIMMs are installed

V2500 memory configurations

In the V2500 server, Excalibur Pluggable Memory Boards (EPMBs) are installed in 16 DIMM connectors on the EWMBs.

A V2500 memory board is organized by quadrants, rows, and buses. Each memory board has four quadrants, four rows and eight buses.

The following terms are used to describe a V2500 memory board, as shown in Figure 39:

Slot	The physical location into which DIMMs are installed. There are 16 DIMM slots, each with a unique designator which denotes the slot's quadrant and bus.
Quadrant	A group of four DIMM slots staggered across the memory board.
Buses	Eight buses span the four rows. Each DIMM in a quadrant is on a different bus.
Rows	Each DIMM has SDRAMs on each side and represents two rows. For instance, the first DIMM installed in the system would represent row 0 bus 0 and row 1 bus 0. All DIMMs have the same SDRAMs on both sides. Therefore, rows 0 and 1 will have the same SDRAM size. Rows 2 and 3 will have the same SDRAM size. Bus interleaving can be configured to either 4 way or 8 way bus interleaving. 8 way provides the best performance. To achieve 8 way bus interleaving, all buses on a row must be populated with DIMMs having the same SDRAM size.

Table 70 shows the correlation between a DIMM slot and a row bus intersection. The first DIMM to be installed in a memory board, Q0B0, occupies row 0 bus 0 and row 1 bus 0 in quadrant 0.

Table 70 **DIMM row/bus table**

Rows	Buses							
	0	1	2	3	4	5	6	7
0	Q0B0	Q0B1	Q0B2	Q0B3	Q1B4	Q1B5	Q1B6	Q1B7
1								
2	Q2B0	Q2B1	Q2B2	Q2B3	Q3B4	Q3B5	Q3B6	Q3B7
3								

V2500 DIMM quadrant designations

Memory boards can be populated in increments of four DIMMs called quadrants.

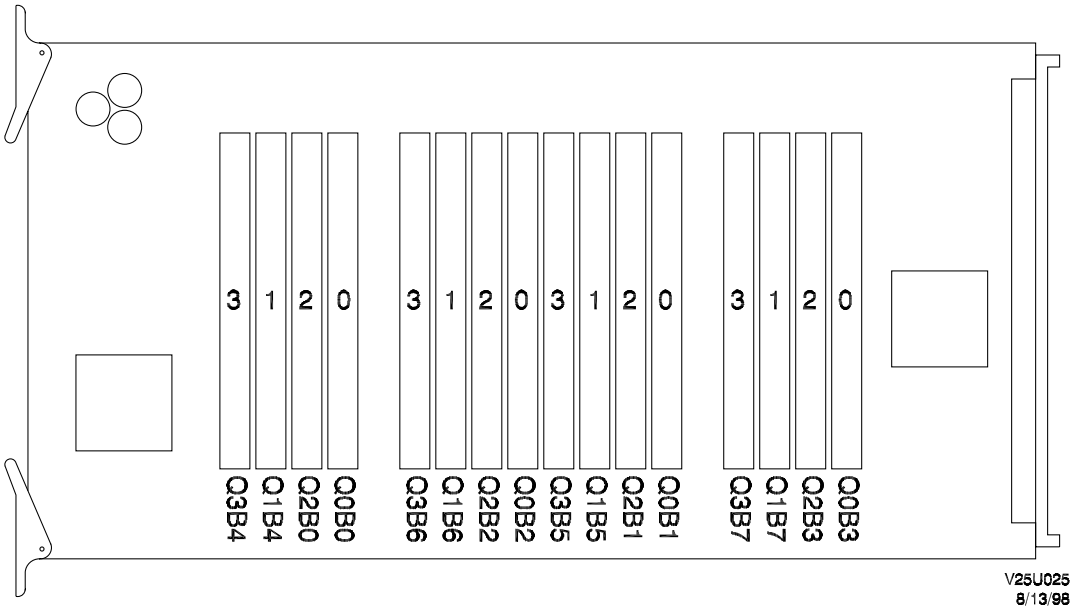
- Four DIMMS provides 1/4 population
- Eight DIMMS provides 1/2 population
- Twelve DIMMS provides 3/4 population
- Sixteen DIMMS provides full population

Table 71 shows the rows and buses associated with each quadrant ID and Figure 39 shows how these are laid out on the memory board.

Table 71 **Quadrant assignments**

Rows	Buses							
	0	1	2	3	4	5	6	7
0	Quadrant 0				Quadrant 1			
1								
2	Quadrant 2				Quadrant 3			
3								

Figure 39 V2500 DIMM locations



Example:

Q2B3: Quadrant 2, Bank 3

V2500 DIMM configuration rules

Use the following rules to plan the memory board DIMM configuration:

- All memory boards must be populated identically.
- Single node memory boards may be populated in 1/4, 1/2, 3/4, or full increments.
- Multi node memory boards may be populated in only 1/4, 1/2, or full increments.
- All DIMMs within a quadrant must be of the same size: 32 Mbyte, 128 Mbyte or 256 Mbyte.
- DIMMs in quadrant 0 can be of a different size than DIMMs in quadrant 2 or 3 without degrading performance.

- DIMMs in quadrant 1 can be of a different size than DIMMs in quadrant 2 or 3 without degrading performance.
- DIMMS in quadrant 0 and 1 should be the same size for maximum performance.
- DIMMS in quadrant 2 and 3 should be the same size for maximum performance.
- DIMMs in quadrant 0 can be of a different size than DIMMs in quadrant 1. To allow this memory to be fully utilized, the bus interleave span will be reduced to 4 way bus interleaving. This will degrade performance.
- DIMMs in quadrant 2 can be of a different size than DIMMs in quadrant 3. To allow this memory to be fully utilized, the bus interleave span will be reduced to 4 way bus interleaving. This will degrade performance.
- Mixing of 32-Mbyte DIMMS and 256-Mbyte DIMMs is not supported.
- All quadrants on a given memory board do not have to be populated with DIMMs.

V2500 memory board configuration rules

The V2500 system supports up to eight memory boards. Valid configurations of memory boards include two, four, and eight. (A six memory board configuration is not supported.) The first two memory boards, as shown in Table 72 on page 173, are located in slots MB0L and MB1l.

Table 72 Memory board configurations

Order	Slot locations
Minimum system configuration	MB0L MB1L
Four memory boards	MB6R MB7R
Eight memory boards	MB2R MB3R MB4L MB5L

User parameters

The Test Controller allows `mem3000` 20 user parameters. Table 73 defines these parameters:

Table 73 User parameter definitions

Words	Usage
0/1	64-bit user pattern 0 used in subtests 238 and 338 (defaults=0xa5a5a5a5/0xa5a5a5a5)
2/3	64-bit user pattern 1 used in subtests 238 and 338 (defaults=0x5a5a5a5a/0x5a5a5a5a)
4	Denotes 88-bit DIMMs are installed (default=2)
5	Denotes test is to run with errors disabled (default=0)
6/7	Octant mask. (default: 0xffffffff 0xffffffff)

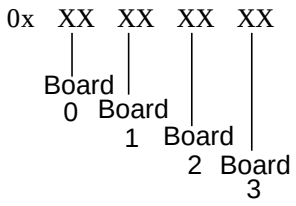
Parameter 4 defaults to the value 2 causing the test to automatically probe all known DIMMs to determine their type: 80- or 88-bit DIMMs. The test then changes the parameter from 2 to 0 or 1. It is set to 1 if only 88-bit DIMMs were found. If any 80-bit DIMMs were found, it is set to 0.

Parameters 6 and 7 default to the value 0xfffffffff, the bit mask that indicates whether a memory octant should be tested. When the Test Controller is started, `mem3000` changes the values to match the memory that `POST` enabled on the node.

Each range is 0x0 through 0xffffffff. Each byte represents the physical octant mask for a memory board.

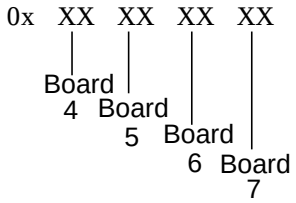
Parameter 6 contains the masks for boards 0 through 3 in the order shown in Figure 40.

Figure 40 Format of parameter 6



Parameter 7 contains the masks for boards 4-7 in the order shown in Figure 41.

Figure 41 Format of parameter7



As an example, the Octant Mask for board 0 is encoded in the first two digits of Parameter 6.

Subtests 100, 101, 150, and 310-338 DO NOT use the Octant Mask. Subtests 100 and 101 test CSRs on all enabled SMACs. Subtests 150 and 310-338 use the Main Memory Map built by POST.

mem3000 error codes

When a failure is encountered, an event code is set along with an error message. The least significant 12 bits of the event code contain the error code. Table 74 lists the `mem3000` error codes.

Table 74 `mem3000` error codes

Code	Meaning
001	Diagnostic address CSR miscompare occurred (upper 32-bits)
002	Diagnostic address CSR miscompare occurred (lower 32-bits)
003	Diagnostic data CSR miscompare occurred (used only by class 1)
004	Diagnostic data CSR miscompare occurred (in upper 32-bits)
005	Diagnostic data CSR miscompare occurred (in lower 32-bits)
008	Miscompare occurred in the upper 32-bits of the CSR
009	Miscompare occurred in the lower 32-bits of the CSR
010	Memory data miscompare occurred
011	Memory data miscompare occurred (upper 32-bits)
012	Memory data miscompare occurred (lower 32-bits)
013	Memory data matched when it shouldn't have (upper 32 bits)
014	Memory data matched when it shouldn't have (lower 32-bits)
020	Miscompare occurred in the upper 32-bits of the tag
021	Miscompare occurred in the lower 32-bits of the tag
022	The tag changed when it shouldn't have
030	ECC data miscompare occurred
031	An ECC error was logged when it shouldn't have been
032	SMAC did not correct the single bit ECC failure as expected

Code	Meaning
033	SMAC did not log the occurrence of a single bit ECC failure
035	SMAC did not log the occurrence of a double bit ECC failure
040	Data miscompare error occurred in sequence #1 of MarchC test (upper 32-bits)
041	Data miscompare error occurred in sequence #1 of MarchC test (lower 32-bits)
042	Data miscompare error occurred in sequence #2 of MarchC test (upper 32-bits)
043	Data miscompare error occurred in sequence #2 of MarchC test (lower 32-bits)
044	Data miscompare error occurred in sequence #3 of MarchC test (upper 32-bits)
045	Data miscompare error occurred in sequence #3 of MarchC test (lower 32-bits)
046	Data miscompare error occurred in sequence #4 of MarchC test (upper 32-bits)
047	Data miscompare error occurred in sequence #4 of MarchC test (lower 32-bits)
060	A semaphore operation did not trigger
070	Incorrect data returned for a semaphore operation
080*	Incorrect info in SMAC error CSRs (single bit data ECC - read)
090*	Incorrect info in SMAC error CSRs (single bit tag ECC - read)
0a0*	Incorrect info in SMAC error CSRs (double bit data ECC - read)
0b0*	Incorrect info in SMAC error CSRs (double bit data ECC - coh_inc op)
0c0*	Tag state did not equal ERROR as it should have

Code	Meaning
0d0*	Tag state did not equal INVALID as it should have
0e0*	An unexpected error was detected in the SMAC error CSRs
100*	Uninstalled Memory
110*	Invalid CSR
120*	Network Cache
130*	Unprotected Memory
140*	Alternate Interleave
150	An HPMC was detected on access to the specified address
200	Denotes the EWMB contains all 80-bit DIMMs
201	Denotes the EWMB contains all 88-bit DIMMs
202	Denotes the EWMB contains a mixture of 80-bit and 88-bit DIMMs
220	Some portions of test code is copied to memory and branched to in attempt to load the code into the icache.The initialization routine detected that code failed to implicitly encache when executed from coherent memory.

The asterisks next to the error codes listed in Table 74 actually indicate a range of events as shown in Table 75.

Table 75 Extended range for error codes

Code	Meaning
code+1	Error cause CSR miscompare error (upper 32-bits)
code+2	Error cause CSR miscompare error (lower 32-bits)
code+3	Error info CSR miscompare error in the err type field
code+4	Error info CSR miscompare error in the ENUM field
code+5	Error info CSR miscompare error in the cc/msg field

Code	Meaning
code+6	Error address CSR miscompare error (upper 32-bits)
code+7	Error address CSR miscompare error (lower 32-bits)
code+8	Error info CSR syndrome code miscompare error

Table 76 Patterns used in specified subtests

Subtest	Pattern
230/330	0x7f7f7f7f7f7f7f7f and 0x8080808080808080
231/331	0xbfbfbfbfbfbfbfbf and 0x4040404040404040
232/332	0xdfdfdfdfdfdfdfdf and 0x2020202020202020
233/333	0xefefefefefefefef and 0x1010101010101010
234/334	0xf7f7f7f7f7f7f7f7 and 0x0808080808080808
235/335	0xfbfbfbfbfbfbfbfb and 0x0404040404040404
236/336	0xfdfdfdfdfdfdfdfd and 0x0202020202020202
237/337	0xfefefefefefefefe and 0x0101010101010101
238/338	0xa5a5a5a5a5a5a5a5 and 0x5a5a5a5a5a5a5a5a (user parameters 0-3)

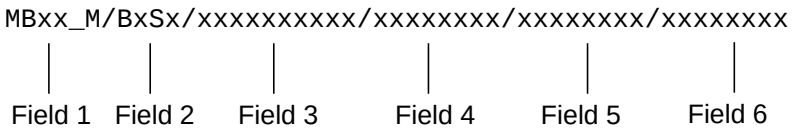
Error messages

When a failure is encountered an event code is set along with an error message. The least significant 12 bits of the event code contain the error code. The error codes and their error message descriptions are defined in the following section. Error codes can have one of three different formats.

Type one error format

Type one errors are used by many of the subtests. Figure 42 shows the format of the type one error format.

Figure 42 **Type one error message format**



There are six fields separated by / symbols. The meaning of each field is as follows:

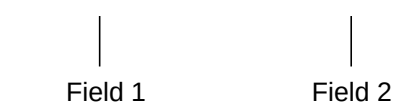
- Field 1—Specifies the on which the failure was detected
- Field 2—Specifies the DIMM on which the failure was detected
- Field 3—Specifies the failing 40-bit address
- Field 4—Specifies the actual 32-bits of data
- Field 5—Specifies the expected 32-bits of data
- Field 6—Specifies the error as follows:
  - COH-OP—Coherent operation
  - DCSR—Diagnostic CSR access
  - CSR DATA - CSR data mismatch
  - DECC— ECC mismatch
  - DTAG—TAG mismatch
  - DDAT—DATA mismatch

Type two errors

The type two error is used only by subtest 640 which determines what type of DIMMs are installed on the first EWMB specified.

A type two error is shown Figure 43.

Figure 43 **Type two error message format**



The two fields of the type two error are as follows:

- Field 1—Specifies the EWMB to which the information pertains
- Field 2—Specifies the type of DIMM detected as follows:
  - x—Non-existent DIMM
  - 0—80-bit DIMM
  - 1—88-bit DIMM

The correspondence of these values to the actual DIMM locations is shown in Figure 44.

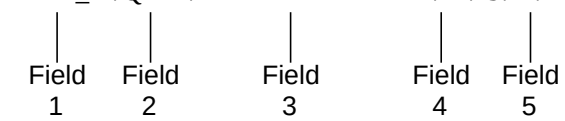
Figure 44 Corresponding type two values to DIMM location

```
   Q0:xxxx Q1:xxxx Q2:xxxx Q3:xxxx
Bus:  0123    4567    0123    4567
```

Type three errors

The type three error (shown in Figure 45) is used only by class 3 subtests to report spurious single-bit ECC errors that occur during testing. The test is designed specifically to bring out these types of failures. However, if failures of other types occur, they are reported in their respective format.

Figure 45 Type 3 error message format



There are five fields separated by / symbols. The meaning of each field is as follows:

- Field 1—Specifies the EWMB on which the failure was detected
- Field 2—Specifies the DIMM on which the failure was detected
- Field 3—Specifies the SMAC error address CSR value
- Field 4—Specifies the syndrome bits
- Field 5—Reminds that this is a single bit ECC error

Notes on mem3000

There is a dependency upon POST to initialize the memory system. This test uses many of the CSR values from POST and does not reconfigure the system. There are some exceptions in which CSR values need to be changed in order for the test to run. In these cases, CSR values should be returned to their previous value upon successful completion of the subtest. If a failure occurs, these CSRs may not be returned to their pre-test state in an attempt to save the failing state configuration.

`mem3000` currently uses the following algorithm for selecting processors to be used in testing: A list is made of processors. Even numbered processors under 16 are first, then odd numbered processors under 16, followed by even then odd CPUs over 16. This ordered list is then used to assign one processor per memory board.

The EWMBs must be installed in pairs (1 even for each odd). Three pairs is not a valid configuration and POST will hardware deconfigure the extra pair. Therefore, either 2, 4, or 8 EWMBs must be installed.

mem3000 uses memory that was enabled by POST to do the pre-test initialization and encaching. Therefore, the Octant Mask parameters (6 and 7) are ignored during subtest init. As a result, lines that are not tested may be re-initialized and used during the encaching sequence.

Subtest 150 and the class 3 subtests use memory interleave and thus test over a range of EWMBs. The memory tested is that which was enabled by POST in the Main Memory Map.

Subtest 150 and the class 3 subtests use coherent accesses to test consecutive memory lines which are interleaved across EWMBs, buses and banks. As a result, parameters 6 and 7 are ignored. When a failure occurs, the failing 40-bit address can be used to determine which logical row, bank, bus, and board was being accessed. The failing DIMM field (QxBx) takes interleaving into account and reports the actual physical Quadrant and Bus that failed.

Depending on the configuration, subtest 640 may not be able to test all EWMBs in the node at once. If subtest 640 does not report the status of all EWMBs the first time, deselect the EWMBs that were tested and rerun the subtest.

Subtests in class 6 will produce HPMCs (indicated by the Test Controller printing the # character). These are expected.

10 Scan test

The Exemplar scan test (`est`) is a diagnostic utility that uses the system scan hardware making it possible to perform connectivity tests and to test gate array internal registers. The `est` utility runs on the teststation and sends scan instructions to a given node by way of the Ethernet.

est utility test environment

est is started on the teststation and is located in /spp/bin/est. The user has the option of either starting up a user interface or having the est utility run a script.

est works on one node at a time by sending scan instructions and data and receiving the results over the diagnostic ethernet connection.

Since est has to communicate closely with the Utilities board, no other diagnostic can be run at the same time. Also, while est is moving data through the scan rings, the operating system can not be running.

est works on the JTAG scan rings throughout the system. Tests provided are:

- Ring (test command r)—Moves data through the scan rings to make sure the rings are connected and that basic scan hardware is operational.
- Dc connectivity (command d)—Checks that wires on the boards between scan devices are intact (no shorts or opens).
- Ac connectivity (command a)—Examines wires on the boards. Ac tests look for timing problems between parts at full speed. If dc connectivity patterns passed, but ac connectivity failed, the failure is bound to be timing related.
- Gate array (command g)—Executes scan tests internal to selected arrays. When these tests fail, the array usually has to be replaced.

These tests are listed in the order in which they should normally be run.

Control of utility board

To prevent unexpected shutdowns from hardware that is sensitive to scan operations, est takes control of a power signal on the Utility board. To control this signal, est must freeze some of the bits of the Utilities board. Therefore, when est starts, it automatically performs the id_verify operation and ring test (command r) on the Utility Ring (ring 22). est then locks the bits to control the power signal. If the user needs to run the id_verify or ring test function, scan operations will occur in all scan rings except the Utility Ring.

To perform ID and ring checks in the utility system, the user should turn off the power control feature either though the command line argument `-p` or through a runtime option command (`power_control`). The latter should seldom occur, because `est` automatically runs these tests on the utility scan path at start up and reports any errors found.

est exit and reset

To quit, `est` calls a script called `est_exit`. The default script performs a `do_reset` function (see “do_reset” on page 284) to reset the node under test. When the CTI cables are tested, `est` directs `est_exit` and `do_reset` to reset the entire complex. To accomplish this reset, `est` passes to the `est_exit` script a parameter that indicates which node of the complex to reset. The script then hands this parameter to `do_reset` which then performs the reset operation.

The default script resides at /spp/scripts/est_exit. If the user wishes to run his own version, he should create the file in a local subdirectory ./scripts/est_exit. If est sees such a file, it will runs the local copy instead of the default. The purpose of the default `do_reset` function is to make sure that the utility system is restored in order to monitor environmental conditions.

est user interfaces

`est` can be run from either a GUI or a command line interface. The est GUI is described in “Running the est GUI” on page 186. The command line interface is described in “Running est from command line” on page 200.

Running the est GUI

The `est` GUI may be started at the command prompt. The following is the `est` command usage:

```
/spp/bin/est [-option] node_number
```

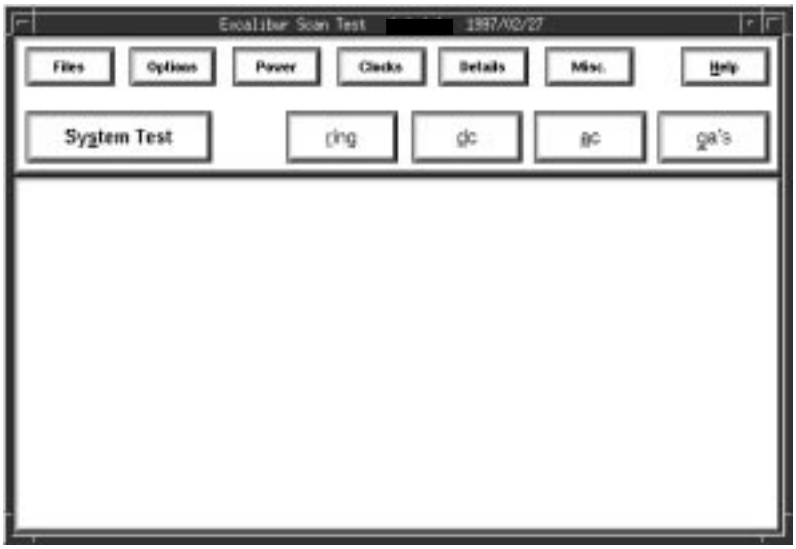
As an example to bring up the GUI and test node 0, enter the following command:

```
% /spp/bin/est -x 0
```

Table 77 on page 200 provides a complete list of options.

Figure 46 shows the `est` main window.

Figure 46 **`est` main window**



The main window has two sections. The upper section has two rows of buttons. The top row provides the user several options to control system and test parameters, and the bottom row allows the user to run all available tests. The lower section is the main window pane that displays messages and test status.

The lower set of buttons allows the user to quickly and easily run the scan tests in a wholesale fashion. The test can be modified to run fewer patterns, to loop continuously or for a finite number of times, to test non-default limits, etc.

Each button is explained in the following sections.

System Test button

Clicking the System Test button runs each set of tests in the following order: ring tests, dc connectivity tests, ac connectivity tests, and gate array tests. It is equivalent to entering the `r`, `d`, `a`, and `g` commands from the command line interface.

ring button

Clicking the ring button runs the system scan tests on all rings and scan paths using the default patterns. It is equivalent to entering **`r`** from the command line interface. Most scan rings are defined in the IEEE 1149.1 JTAG specification.

dc button

Clicking the dc button runs only the dc connectivity tests using default parameters. It is equivalent to entering **`d`** from the command line interface.

ac button

Clicking the ac button runs only the ac connectivity tests using default parameters. It is equivalent to entering **`a`** from the command line interface.

ga's button

Clicking the ga's button runs only the gate array tests using default parameters. It is equivalent to entering **`g`** from the command line interface. When the Limit Test Patterns option is set in the Options window, however, clicking the ga's button runs the gate array tests with the limited number of patterns specified. See "Options button" on page 188.

Files button

Clicking the Files button opens pop-up menu with three selections:

- Execute Scripts—Runs a file containing `est` commands.
- Reset Log File—Clears the log file.
- Exit—Closes the est main window and exits the program.

Options button

Clicking the Options button opens pop-up menu with seven selections:

- Log_File—Generates a log file and stores it in /spp/data/est.log.
- Stop On Error—Causes the test(s) to halt whenever an error is detected.
- Limit Test Patterns—Limits the number of test patterns so that the test runs in approximately one-half the normal time. Test coverage drops to approximately 90%.
- Limit Error Report—Limits the length of the error report to 10 errors.
- Normal Font Size—Prints status to the main window pane using the standard font size.
- Large Font Size—Prints status to the main window pane using a large font size.
- Show time—Prints current time and date.

Power button

Clicking the Power button opens pop-up menu with four selections:

- Upper—Sets the upper limit of the power supplies.
- Nominal—Sets the power supplies to their nominal values.
- Lower—Sets the lower limit of the power supplies.
- Status—Displays the current settings of the power supply voltages (upper, normal, or lower). When this option is invoked, it displays both the power supplies and clock settings.

Clocks button

Clicking the Clocks button opens pop-up menu with four selections:

- Upper—Sets the upper limit of the system clocks.
- Nominal—Sets the system clocks to their nominal values.
- External—Selects an external clock from the ECUB.
- Status—Displays the current settings of the power supply voltages (upper, normal, or lower). When this option is invoked, it displays both the clock and power supplies settings.

Details button

Clicking the Details button opens a pop-up menu with seven selections:

- P/F Each Pattern—Displays the number and the results of each pattern in the test.
- Test File Msgs—Prints the pattern file, both instructions and data. This option is primarily used for troubleshooting `est`.
- Show Scan Instr—Displays the instruction portion of the scan packet. This option is primarily used for troubleshooting `est`.
- Show Scan Data—Displays the data portion of the scan packet.This option is primarily used for troubleshooting `est`.
- Show SDP Data—Displays the scan data protocol portion of the scan packet. This option is primarily used for troubleshooting `est`.
- Show GUI Commands—Displays each command in the main window pane. This option is useful for writing test scripts.
- Enable GUI Commands—Toggles between enabling and disabling GUI commands. This option is used with the Show GUI option to assist in writing test scripts.

Misc. button

Clicking the Misc. button opens pop-up menu with nine selections:

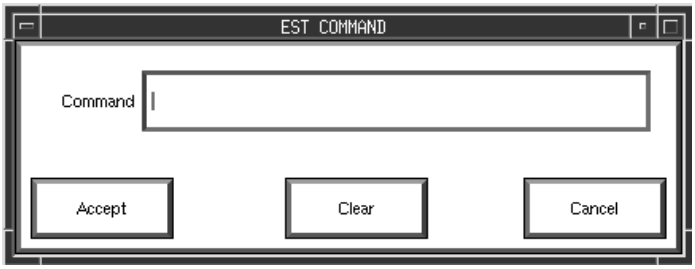
- Goto Safe State—Places the system hardware in a safe state
- Verify Config—Compares returned test scan data from the JTAG interface against the configuration file.

- Command Menu—Opens the command line window which allows the user to enter `est` commands directly from the GUI system.
- Scan Debug Menu—Opens the debug window.
- Connectivity Test Menu—Opens the connectivity test window.
- Gate Array Test Menu—Opens the gate array test window. Gate array tests use test vectors that have been generated for the certain arrays (each array has multiple files associated with it).
- Sci Test Menu—Opens the SCI test window. The tests verify the Coherent Toroidal Interface (CTI) cables between nodes.
- Abort—Stops the currently running test

Command line window

The `est` command line window allows the user the freedom to enter a command directly from the `est` GUI system. Figure 47 shows the `est` command line window.

Figure 47 **`est` command line window**

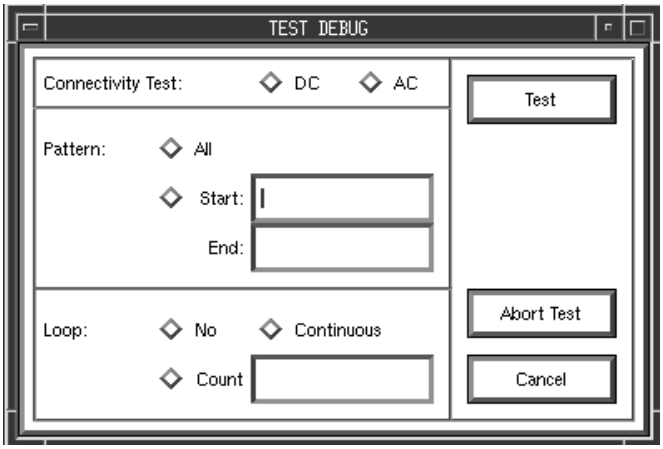


To issue a direct command, click in the Command field, enter the command and then press the **Return** key. `est` executes the command with output going to the main window. Clicking the Accept button repeats the command. The Clear button clears the command line. Clicking the Cancel button closes the window.

Connectivity test window

The connectivity test window invokes the connectivity tests. With this window, the user can select either the ac or dc test, the starting, ending, or all patterns, and the test looping parameters. Figure 48 shows the `est` connect window.

Figure 48 **est connectivity window**



To select a connectivity test, click on either the dc or ac button in the Connectivity Test panel.

In the Pattern panel, clicking the All button runs each test pattern. est creates the patterns on the fly based on the number of testable wires in the system. The user can also select the starting and ending patterns by clicking the button next to the start field. Enter the appropriate data in the Start and End fields. The Start and End options are normally used when debugging a system or board.

The Loop panel has three check buttons:

- No—Disables looping.
- Continuous—Enables continuously test looping.
- Count—Enables test looping a finite number of times. To set the number of times the test loops, click the Count button and enter the number of loops in the Count field.

To start the test, click the Test button; to stop it, click the Abort Test button.

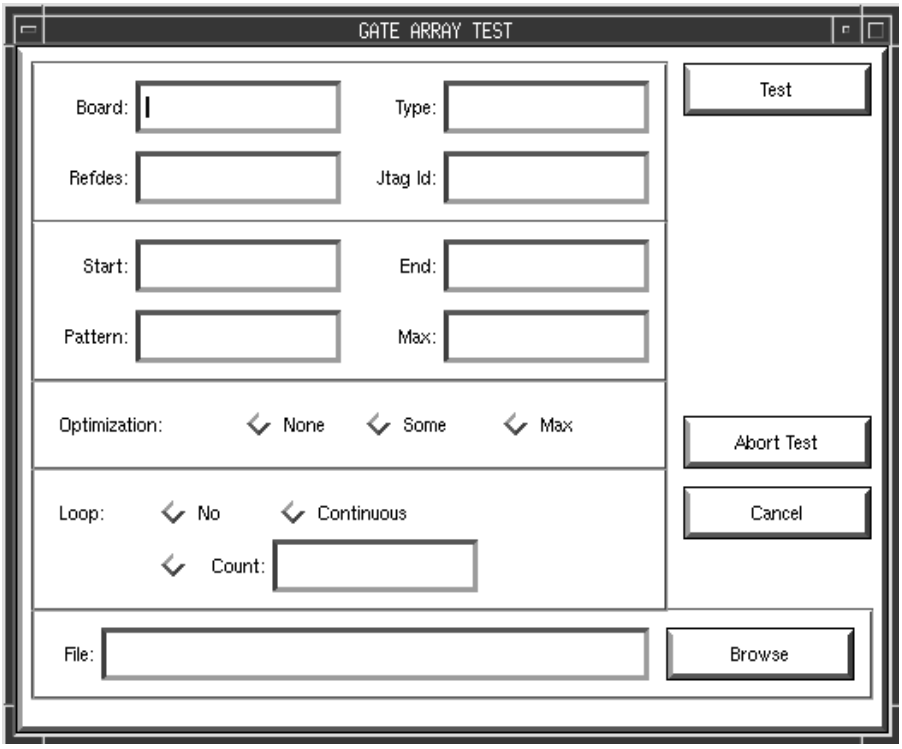
Clicking the Cancel button closes the connectivity window.

Gate array test window

The gate array test window provides a means to test all gate arrays in the Exemplar system. The window is simple to use.

Figure 49 shows the `est` gate array test window.

Figure 49 **`est` gate array test window**



In the top panel, enter the following data in the appropriate fields:

- Board—Sets the location of the gate array.
- Type—Sets the type of gate array.
- Refdes—Sets the reference designation of the gate array.
- Jtag—Sets the associated JTAG identification.

When more than one field is used, `est` picks what to test by ANDing the fields.

The next lower panel determines which and how many patterns are used in the gate array test. The test normally uses all patterns, but, for troubleshooting, you may set the starting and ending patterns, set the maximum number of patterns (a range of patterns), or set a single, custom pattern. Enter the following test pattern information in the appropriate fields:

- Start—Sets the starting pattern.
- End—Sets the ending pattern.
- Pattern—Sets a custom pattern.
- Max—Sets the range of patterns to be used in the gate array test. The default is to use all patterns.

In the next lower panel, click the appropriate test optimization buttons:

- None—No optimization.
- Some—Increased optimization.
- Max—Maximum test optimization.

The next lower panel controls the looping parameters:

- No—Disables looping. The test is only run once.
- Continuous—Enables continuous test looping. When running in continuous looping, the test is halted by clicking the Abort Test button.
- Count—Enables controlled looping. The number of loops is entered in the Count field.

The gate array test window may also be loaded with predefined parameters file. To load a file, click the Browse button and locate the appropriate file in the browse window.

Clicking the large buttons in the gate array test window has the following effect:

- Test—Starts the gate array test.
- Abort Test—Stops the test.
- Cancel—Closes the gate array test window.

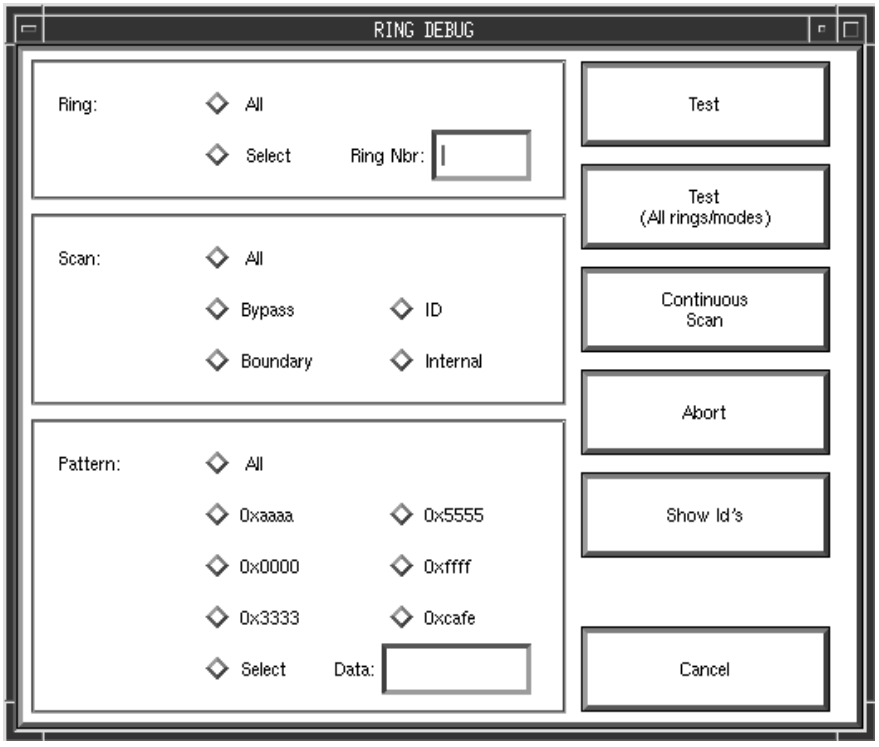
Scan window

The scan window provides means of testing the system scan rings. Figure 50 shows the `est` scan window.

NOTE

For more information on scan rings and modes, see the IEEE 1149.1 JTAG specification.

Figure 50 **est scan window**



The window has three panels: Ring, Scan, and Pattern.

Clicking the buttons in the Ring panel has the following effect:

- All—Tests all available rings in the system.
- Select—Allows the user to test a particular ring by entering the ring number to be tested in the Ring Nbr field.

Clicking the buttons in the Scan panel sets the scan paths. All scan modes can be selected or the test can be set up to test the individual pathways as follows:

- All—Tests all scan modes.
- Bypass—Test the bypass ring.
- ID—Tests JTAG identification ring.
- Boundary—Tests the ring boundary.
- Internal—Test the internal ring.

In the Pattern panel, clicking the All button causes the test to use all available patterns. Clicking the button next to a particular pattern causes the test to only use that pattern (plus any others that are checked at the same time). Clicking the Select button allows the user to specify the test pattern by entering it in the Data field.

Clicking the large buttons in the scan test window has the following effect:

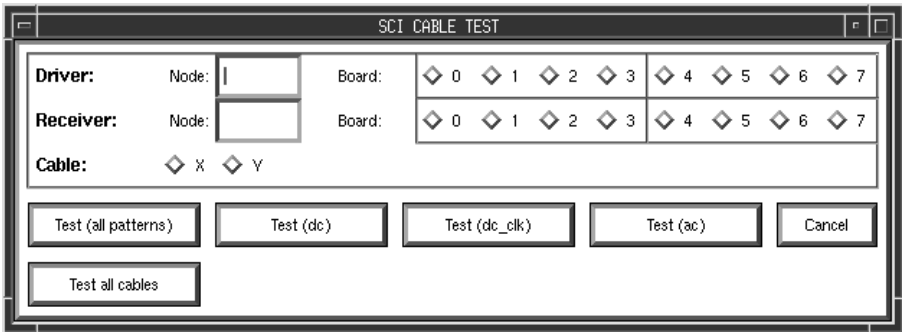
- Test—Starts the scan test. The rings, scan parameters and patterns selected in the scan window are invoked by this button.
- Test (All rings/modes)—Starts the scan test using all rings and patterns regardless of what is selected in the scan window.
- Continuous Scan—Places the scan test in continuous looping.
- Abort Test—Stops the test.
- Show Id's—Shows the JTAG IDs of all devices in the appropriate scan rings.
- Cancel—Closes the scan test window.

SCI cable test window

The SCI cable test window provides a means to test the cables that connect the scalable coherent interfaces between nodes. All cables are tested by default, but an individual cable can be tested using this window.

Figure 51 shows the `est` SCI cable test window.

Figure 51 **est SCI cable test window**



In the top panel are two rows of fields and buttons that determine source port (Driver) of the cable and the destination (Receiver). A third row selects either the X or Y cable. For both the Driver and Receiver select a node and EMB. Enter the desired node number in the node field. Click the interface number (0 through 8). Click either or both the X-ring cable or Y-ring cable.

The buttons in the lower portion of the window have the following effect:

- Test (all patterns)—Runs the SCI cable test using all test patterns.
- Test (dc)—Performs the continuity test on the cable.
- Test (dc_clk)—Performs the continuity test on the cable clock lines.
- Test (ac)—Performs dynamic test on the cable.
- Test all cables—performs full cable test suite on all interface cables.

CAUTION

Before running the full cable test suite, refer to “SCI_all test” on page 208.

- Cancel—Closes the window.

Help

Clicking the Help button opens pop-up menu with five topic selections:

- Overview
- Commands
- GUI
- Input Files
- Options

Clicking on one of these options opens the Help window shown in Figure 52. This window is initially blank.

To open the topic of interest, click the Browser button. This opens the Help browser window shown in Figure 53. Double click on a topic listed in the browser.

Figure 52 **est Help window**

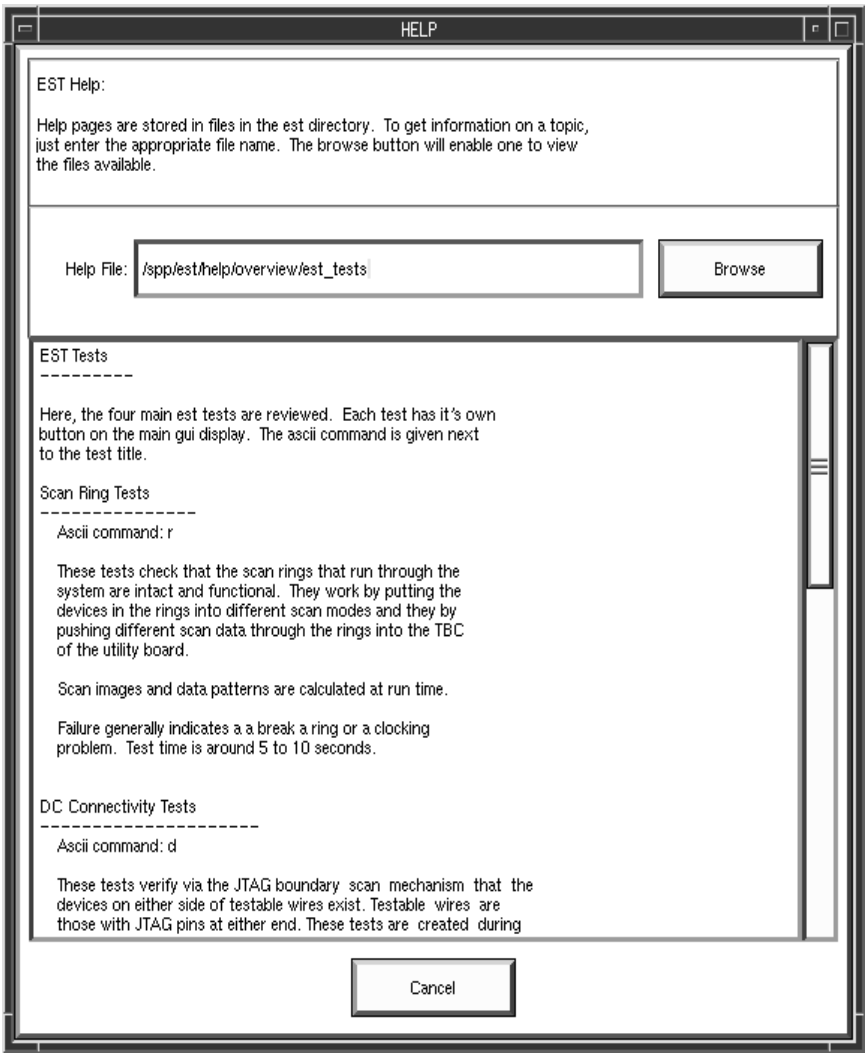
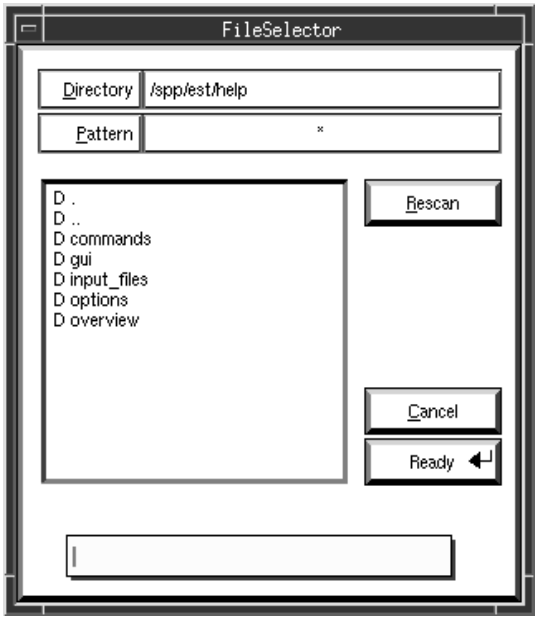


Figure 53 **est Help browser window**



Running est from command line

The following is the command line usage for est:

```
est [-options] <node_number>
```

For example, to test node 0, enter:

```
% est 0
```

est reads configuration information from files stored in /spp/data (e.g node_0.cfg). These configuration files are automatically generated by ccmd each time the system is powered up. While ccmd is running, it prints its status to the console window. When database generation is complete and no errors are reported, it writes the necessary configuration files and est may be executed.

Table 77 shows est command line options.

Table 77 est command line options

Option	Description
-v	Print version and exit
-f <filename>	Run a given script file
-l	Do not generate a log file
-o <filename>	Redirect the log file to the given filename
-x	Open X windows GUI interface
-y	Say “yes” if asked to take over a locked ECUB
-C	Use old style configuration formats
-V	VT100 command menu
-A	Standalone; does not connect to node
-B	Do not build database
-H	Hardware mode on (default)
-N	Hardware mode off

Option	Description
-P	Do not let est handle the MIB power control
-U	<on | off> UTS support option
-Y	Force est_config to be run
-Z	Force est_config not to be run

Some examples of est usage are:

```
est -v
```

```
est -l -f my_script 0
```

```
est -o ./my_log_file 0
```

The est utility uses certain data and vector files located in the /spp/est directory.

Unless disabled or redirected, the est utility will generate a log file, est.log, and store it /spp/data/est.log. Any previous log file will be renamed to est.log.old

Running `est` from command line

Example of output when `est` is started:

```
% est 0

Excalibur Scan Test     1.0.0.2  1998/11/25 10:32:58   Steven Terry
.......................
.....
General EST Tests:
c       ... compare id's to config file
r       ... scan ring test
d       ... board level dc tests
a       ... board level ac tests
g [options] [file] ...  gate array tests

Special Scan Tests:
b       ... bypass/id test
i       ... print id's found in design

EST Options:
F       ... set option & debug flags
q       ... quit nicely, ask first
qq      ... quit nicely, don't ask
Q       ... quit, not so nice
h       ... print this help message
v       ... print EST version info
!cmd    ... send the command to Unix (ex. "!ls patterns")
>>
```

Example output when using the `est -h` option:

```
% est -h
Excalibur Scan Test    1.0.0.2 1998/11/25 10:32:58   Steven Terry

usage: est [-options] [server] node [-cp port] [-sp port]
options:
        -h ... print this help message
        -v ... print the version of the program and exit

        -l ... turn OFF log file for this session
        -f <file> ... get commands from <file>
        -o <file> ... redirect log file to <file>

        -x ... X windows gui interface
        -y ... say "yes" if asked to take over a locked ecub
        -C ... use old style config file formats
        -V ... vt100 command menu

        -A ... stand alone; does not connect to node
        -B ... do not build database
        -H ... harware mode on (default)
        -N ... hardware mode off
        -U <on|off> ... UTS support option
        -Y ... force est_config to be run
        -Z ... force est_config not to be run
        -P ... do not let est handle the midplane's power control

ports:
        -cp ... client port
        -sp ... server port
```

AC Connectivity test

The ac Connectivity test format is:

`a [-s -p #]`

Table 78 shows the options for the this test.

Table 78 **AC Connectivity test options**

Option	Description
-s	Step mode (for debug purposes).
-p <number>	Run pattern number only.

Bypass test

The Bypass test format is:

b

The Bypass test places the scan ring hardware into bypass mode.

DC Connectivity test

DC Connectivity test format is:

d [-s -p #]

Table 79 shows the options for the this test.

Table 79 **Dc Connectivity test options**

Option	Description
-s	Step mode (for debug purposes).
-p <number>	Run pattern number only.

Gate Array test

The Gate Array test format is:

g [options] [pattern file]

Table 80 shows the options for the this test.

Table 80 Gate Array test options

Option	Description
-r <refdes>	Test arrays with matching reference designator value.
-b <board>	Test arrays on given board. <board> may either be a number or a name.
-j <jtag_id>	Test arrays matching a jtag_id.
-t <type>	Test an array type (For example, ERAC).
-s <number>	Start with a given pattern number.
-e <number>	End on a certain pattern number.
-m <number>	Run a maximum of <number> patterns per file.
-o <number>	Optimization level (0, 1, or 2) Two is the most optimized and is the default.

By default, the g command tests all arrays. When the -r, -b, -j, or -t options are used, only arrays that meet all criteria are tested.

Gate array tests use test vectors that have been pregenerated for the certain arrays (each array has multiple files associated with it). The -s, -e, and -o options can be used to limit the number of patterns that are run for each pattern file. The default is to run all patterns in each file. While it may take more time to run all patterns, using options to limit the number of patterns may result in a significant loss of test coverage.

The -o option controls how much parallelism takes place. There are three levels:

- 0—No optimization.
- 1—Same parts on the same ring are tested together.
- 2—Identical rings are tested in parallel.

If an error is encountered during parallel testing, the following message may appear:

```
*** errors found - must switch to serial testing ***
```

When an error occurs, parallel scans into the scan hardware may result in bus conflicts on TDO pins. Therefore, `est` automatically stops using parallel scans when errors happen.

SCI test

The sci utility tests the Coherent Toroidal Interface (CTI) cables between nodes. The term SCI (Scalable Coherent Interface) is often used in place of the term CTI; the terms are interchangeable.

The usage of sci is as follows:

```
sci [driver] [receiver] ring test
```

where:

[driver]	Refers to the node and memory board to which the CTI cable is connected and from which the test data originates.
[receiver]	Refers to the node and memory board to which the other end of the CTI cable is connected and receives test data.
ring	Refers to the CTI ring associated with the cable, either x or y.
test	Refers to the specific test: dc, dc_clk, ac. With the dc test, the clock from the receiver node is used. The dc_clk test derives its clock from the cable.

The following is an example of sci usage:

```
% sci 0 mb0l 1 mb0l y dc_clk
```

This command runs the dc_clk test on the cable connected between node 0, memory board 1 (driver) and node 1, memory board 1 receiver.

SCI_all test

The sci_all utility tests all SCI cables in a complex.

The usage of sci_all is as follows:

```
sci_all [test]
```

where:

test	Refers to the specific test: dc, dc_clk, ac. With the dc test, the clock from the receiver node is used. The dc_clk test derives its clock from the cable.

If all cables are not connected or there is an unusual cable configuration, sci_all will not work until the cable configuration files in /spp/est/sci_tests have been updated:

CAUTION Use extreme care when modifying any est data files; damage to the system may result. If you are not sure what to do, seek help.

To run est with modified data files, create a local directory and copy the files to it. For the case of sci_all, create a local ./sci_tests directory and then copy the sci cable files (shown above) to it. Edit the files and then run est. est looks into the ./sci_tests directory for the files before the /spp/est/sci_tests directory.

JTAG Identification test

The JTAG Identification test prints all JTAG IDs. the format is:

```
i
```

Margin commands

The Margin command for clocks and power format is:

```
m [-c | -p [supply] [value]]
```

The -c option specifies the clock, and the -p option specifies power. These two options can not be used together; use either -c or -p.

When a value is not supplied, the current states are displayed.

By itself, m shows all margins.

The following are examples of margin command options:

- **-c high**—Displays the upper clock limit.
- **-p 1 nom**—Sets the supply 1 margin to nominal.

There are four power supplies, 1 through 4.

Table 81 shows the valid values for clock and power.

Table 81 Valid values for clock and power supplies

Clock	Power
up or high	up or high
nom	nom
ext	low

est miscellaneous commands

This section gives the following useful commands entered at the est prompt:

- **ms**—Puts all the scan hardware into a safe state.
- **q**—Quits, but asks the user first.
- **qq**—Quits without asking.
- **script <file>**—Runs a file containing est commands.
- **v**—Prints version information.
- **F**—Opens the flags submenu.
- **t**—Prints current time and date.

est run time option commands

est provides commands that update the main option settings that control the run time operation of est tests. Each command uses one command line argument, on or off. Table 82 lists these commands.

NOTE When in GUI mode, all of the above choices are available through the options and details pull-down menus.

Table 82 **est runtime option commands**

Command	Description	Default argument
log_file	Turn on/off writing to the log file.	On
stop_on_error	Stops the test when an error is detected.	On
limit_patterns	Runs a limited set of patterns when testing arrays. This runs faster, but reduces coverage.	Off
limit_errors	Limits to 10 the max number of errors that will get printed. The total error count is still printed.	On
pass_fail	When enabled, it prints the pass or fail status of each test pattern.	Off
test_file_msgs	Debug option that enables printing of gate array pattern file information.	Off
show_scan_instr	Show scan instructions when running tests.	Off
show_scan_data	Show scan data when running tests.	Off
show_sdp	Show SDP ethernet information when running est.	Off
power_control	Affects whether or not est takes control of the power-down signal in the utility system. Turning this feature off could result in unexpected power shutdowns, but that may be needed for some special debug efforts.	On

est command flags and options

There are a number of flags or options that operate on and enhance the est commands. Some of these flags and options perform the same functions as the run time option commands.

To set these options, enter **F** at the est prompt. This invokes the flags submenu. To exit, press **return** at the flags prompt. This returns the main est prompt.

Some of the more useful options are:

- l—Limits the number of internal array patterns executed by the g command. This has the affect of decreasing coverage to approximately 90 percent.
- s—Stops testing when an error is detected.
- A <number>—Limits the number of ac connectivity tests.

  Setting a limit of zero or less results in all patterns being used.
- D <number>—Limits the number of dc connectivity tests.
- E—Shows SDP packets transferred across the ethernet.
- P—Controls whether or not the pass/fail status of individual patterns are displayed.

Script files

There are two ways of running script files: from the command line (**-f <filename>**) or from the est prompt. From the command line, est executes the instructions listed and when finished, displays the est prompt. To cause est to quit when the script is finished, put **q** at the end of the script file.

The script command reads est commands from an ASCII file and runs those commands. The following rules apply to the file:

- The file must have only one command per line.
- Command syntax must be the same as entered at the est prompt.
- Comments lines must start with a # sign.

An example file might contain the following lines:

```
# check the rings
r
# show pattern pass/fail steps
F P
#limit dc testing to 3 patterns
F D 3
#do dc testing
d
q
```

11 Utilities

This chapter details most of the diagnostic utilities which include:

- address_decode
- arrm
- consolebar
- dcm
- dfdutil
- dump_rdrs
- fwcp
- fw_init
- get_node_info
- hard_logger
- lcd
- load_eprom
- pim_dumper
- set_complex
- soft_decode
- sppconsole
- tc_init
- tc_ioutil
- tc_show_struct
- Version utilities
- Event processing
- Miscellaneous tools

address decode

`address_decode` decodes 40-bit virtual address into the physical node, smac, row, bus, and bank. It has the following format:

```
address_decode <40-bit address in hex>
```

In order to determine the current memory configuration, `address_decode` invokes some `sppdsh` commands to read certain CSR values so that it can take into account the board mapping, row mapping, interleave values, and DIMM sizes present in the system. Consequently, it must be run on a teststation that can access the node via `sppdsh`.

`address_decode` reports an error if the address entered does not exist.

Example of `address_decode` use:

```
% address_decode 0x010f000020
```

In this example, the address decodes the following

- Node ID: 0
- SMAC: 7
- Bus: 3
- Row: 2
- Bank: 1

Example of `itc_ioutil` error:

```
% address_decode 0x01ff000020
This error detected non-existent row!
Decode failure
```

AutoRaid recovery map (arrm)

The arrm utility is used only with an AR-12H (C5447A) disk array that displays the status "No address table" on the front panel rather than the usual status of "Ready." It is only intended for use by trained service personnel in this specific situation.

Starting arrm

To run arrm, enter the followingf command:

tc_ioutil 0 arrm.fw

This script downloads and executes arrm. The following initial display for the operator on the sppconsole window:

Example output of arrm utility

```
**********************************************************************
***                                                                ***
***            AutoRaid Recover Map Utility                        ***
***                                                                ***
***       (C) Copyright Hewlett-Packard Co. 1998                   ***
***                  All Rights Reserved                           ***
***                                                                ***
***  This program may only be used by HP support personnel and     ***
***  those customers with the appropriate Class license or         ***
***  Node license for systems specified by the license.  HP        ***
***  shall not be liable for any damages resulting from misuse     ***
***  or unauthorized use of this program.  This program            ***
***  remains the property of HP.                                   ***
***                                                                ***
***                       Version 4.3                              ***
**********************************************************************
Please enter the address of the AR-12H
  in the form E/S/0.T.0 where
  where E is the EPIC number,
        S is the slot number,
        and T is the target ID number in decimal.
```

For example, if AR-12H at SCSI target ID 5 on a SCSI bus is attached to a controller in slot 1 of EPIC 0, the response would be:

`0/1/0.5.0`

If the EPIC number is outside of the range 0 to 7, the slot number is outside of the range 0 to 2, or the target number is outside of the range 0 to 15, an error message is displayed and the operator prompted to reenter the address. The program then tries to open the path to the array and perform checks of its internal state. If the open and checks are successful, a message similar to the following is displayed:

Example output with sucessful opening

```
Attempting to open 0/1/0.5.0
Checking the Product firmware for RECOVER capability.
Nr.  Vol. Set Ser. Nr   Drives:
-----------------------------------
 1: >   1213A       D<  A4 A5 A6
 2: >   1213A      12<  B4 B5 B6
Please enter number of volume set to recover.
```

Failure to open and recovery

If openning the array fails, it is likely that the address entered for the array was incorrect, and the following message will be displayed:

Example of the Failure on open message

```
Attempting to open 0/1/0.5.0
A failure occurred on openning the array.
Please boot OBP and run the 'show-scsi-all utility to verify the address of the
array and reset the array before retrying. Halting.
```

The `show-scsi-all` utility scans all SCSI buses on the system and displays all SCSI devices found and their addresses.

After successfully openning the path to the array, `arrm` checks internal array states that can affect a successful recovery. If a condition exits that may prevent a successful recovery, `arrm` notifies the operator displays the option whether or not to continue. For example, if the array firmware is of an early revision, the following message will appear:

Example of unsucessful recovery message

Utility Compatibility Check Unsuccessful.
The Product firmware may not support RECOVER!
Do you want to attempt recover anyway ([y]/n)?

In all cases of this type, respond with a **y**, **Y**, **n**, or **N** followed by **ENTER** or just **ENTER**. The default is the choice enclosed in the square brackets (i.e.`[y]`), and just pressing **ENTER** is equivalent to entering the letter enclosed in the square brackets followed by **ENTER**.

The `arrm` utility can only recover one volume set at a time. It displays a list of the volume set(s) present in the array. The display has with one line per volume set. Each line contains the volume set serial number enclosed in angle brackets (>, <) followed by the physical locations of the drives that are members of that volume set. For the above example, the following output shows the unusual case of an array with two volume sets present with three drives in each of the two volume sets.

Example output of recovery volumes

```
Nr.  Vol. Set Ser. Nr   Drives:
-----------------------------------
 1: >   1213A      D<  A4 A5 A6
 2: >   1213A     12<  B4 B5 B6
```

To select the volume set to be recovered, enter the number in the leftmost column of the line describing the volume setand press **ENTER**.

Once the volume set has been selected, the array firmware examines map information on the disks of the selected volume set to determine if recovery is possible. This may take several seconds. If recovery is not possible, the following message will be displayed:

```
Failed -- RECOVER Command
```

This message may also result by attempting a recovery even though the array firmware may not support the recover command.

If recovery is possible, the following message will be displayed:

```
Press ENTER key to exit loop.
```

This message is followed by the following message message

```
Recovery in progress:  xx  %
```

where xx is a number between 0 and 100. This message indicates the percentage of the volume set that has been recovered and is updated approximately once per second. The recovery operation can take several minutes depending on the amount of data in the volume set. To exit the recovery process, press the **ENTER** key.

NOTE Do not exit the recovery process unless the progress indication hangs and does not increment within one or two minutes.

When the recovery completes, the array automatically begins its initialization sequence and the `arrm` utility displays the following messages:

```
Recovery is finished.
Please wait while the array re-initializes.
```

This message is followed in approximately a minute by this message:

```
Array re-initialization is now complete.
```

If more than one volume set is present in the array, the following messages will also be displayed:

```
Multiple volume sets were present.
Please remove the drives belonging to volume sets
that were not recovered and handle them separately.
```

The final message from the arrm utility is:

```
Exiting
```

consolebar

The consolebar utility is an X application that provides a simple interface capable of starting console windows to all V2500 nodes configured on the teststation. It has the following format:

```
consolebar [-display displayname]
```

consolebar retrieves the list of configured nodes and displays the node IDs, grouped by complex. When the push-button for a node is pressed, an xterm is started and the sppconsole program is run against the specified node.

To start consolebar from the teststation root menu, select the consolebar menu item.

To start from a shell (local or remote), ensure that your DISPLAY environment variable is set appropriately before starting consolebar.

For example:

```
$ DISPLAY=myws:0; export DISPLAY     (sh/ksh/sppdsh)
% setenv DISPLAY myws:0              (csh/tcsh)
```

As another example, use the -display start-up option:

```
# consolebar  -display myws:0
```

NOTE For shells run from the Test Station desktop, the DISPLAY variable is set (at shell start-up) to the local teststation display.

dcm

dcm dumps the boot configuration map information for the specified node. There are two main reporting modes; one for general hardware configuration and one for the DIMM type.

The general hardware mode reports processors, ASICs, and memory size information. The DIMM type mode provides pass/fail tests for specific DIMM types, and a general DIMM type report option.

dcm uses following format:

```
dcm [-d <80|88|all>] <node id> <node id> ...
```

-d 80 checks to see if only 80-bit DIMMs are installed.

-d 88 checks to see if only 88-bit DIMMs are installed.

-d all dumps the status of all installed DIMMs, 80- or 88-bit.

This option returns an exit code: a zero value indicates dump was successful and a one values indicates the dump failed.

node id may be a node number or IP name.

When invoked as **dcm <node id>**, dcm returns 0 and prints a table with the following format for a node with eight processors, eight SPACs, one SIOB, and EWMBs half-populated with 128-Mbyte DIMMs:

Output table using `dcm <node_id>`

```
Acquiring Boot Configuration Map...
         Stingray Configuration Map Dump:        Node:  0  (hw2a-0000)
         ============================================================
                 VERSION: 1.0     compiled: 1998/12/16 18:35:00
         CheckSum:0xf407a073
         Boot Config Map Size:164 words
         POST Revision:1.0
CPUs (Rev, ICache, DCache Size in MegaBytes)
============================================
PB0L_A PASS (2.0, 0.50, 1.00)           PB0L_B EMPTY
PB0R_A EMPTY                            PB0R_B EMPTY
PB1R_A PASS (2.0, 0.50, 1.00)           PB1R_B EMPTY
PB1L_A EMPTY                            PB1L_B EMPTY
PB2L_A PASS (2.0, 0.50, 1.00)           PB2L_B EMPTY
PB2R_A EMPTY                            PB2R_B EMPTY
PB3R_A PASS (2.0, 0.50, 1.00)           PB3R_B EMPTY
PB3L_A PASS (2.0, 0.50, 1.00)           PB3L_B PASS (2.0, 0.50, 1.00)
PB4L_A PASS (2.0, 0.50, 1.00)           PB4L_B EMPTY
PB4R_A EMPTY                            PB4R_B EMPTY
PB5R_A PASS (2.0, 0.50, 1.00)           PB5R_B EMPTY
PB5L_A EMPTY                            PB5L_B EMPTY
PB6L_A PASS (2.0, 0.50, 1.00)           PB6L_B EMPTY
PB6R_A EMPTY                            PB6R_B EMPTY
PB7R_A PASS (2.0, 0.50, 1.00)           PB7R_B EMPTY
PB7L_A PASS (2.0, 0.50, 1.00)           PB7L_B PASS (2.0, 0.50, 1.00)
SPACs
=====
0L    -  PASS
P1R    -  PASS
P2L    -  PASS
P3R    -  PASS
P4L    -  PASS
P5R    -  PASS
P6L    -  PASS
P7R    -  PASS
SAGAs
=====
IOLF_B    -  PASS
IOLR_B    -  PASS
IORR_B    -  EMPTY
IORF_B    -  PASS
IOLF_A    -  PASS
IOLR_A    -  PASS
IORR_A    -  EMPTY
IORF_A    -  PASS
SMACs
=====
MB0L_M    -  PASS
MB1L_M    -  PASS
MB2R_M    -  EMPTY
MB3R_M    -  EMPTY
MB4L_M    -  EMPTY
MB5L_M    -  EMPTY
MB6R_M    -  EMPTY
MB7R_M    -  EMPTY
STACs
=====
MB0L_T    -  DECONFIG
MB2R_T    -  EMPTY
MB3R_T    -  EMPTY
MB4L_T    -  EMPTY
```

```
MB5L_T    -  EMPTY
MB6R_T    -  EMPTY
MB7R_T    -  EMPTY
Memory:
=======
    Physical: L=128MB, M=64MB, S=16MB      Logical: l=128MB, m=64MB, s=16MB
    (If logical memory not specified, then it matches physical memory size)

                * = Software Deconfigured      - = Not In Use
EWMB0:
======
EWMB0: Q0B0 S/S          Q1B4 -/-          Q2B0 -/-          Q3B4 -/-
EWMB0: Q0B1 S/S          Q1B5 -/-          Q2B1 -/-          Q3B5 -/-
EWMB0: Q0B2 S/S          Q1B6 -/-          Q2B2 -/-          Q3B6 -/-
EWMB0: Q0B3 S/S          Q1B7 -/-          Q2B3 -/-          Q3B7 -/-
EWMB1:
======
EWMB1: Q0B0 S/S          Q1B4 -/-          Q2B0 -/-          Q3B4 -/-
EWMB1: Q0B1 S/S          Q1B5 -/-          Q2B1 -/-          Q3B5 -/-
EWMB1: Q0B2 S/S          Q1B6 -/-          Q2B2 -/-          Q3B6 -/-
EWMB1: Q0B3 S/S          Q1B7 -/-          Q2B3 -/-          Q3B7 -/-
```

When invoked with **`dcm -d 80 <node id>`**, dcm returns 0 if all installed DIMMs are 80-bit single-node DIMMs. dcm returns a 1 if one or more 88-bit multinode DIMMs are detected.

When invoked with **`dcm -d 88 <node id>`**, dcm returns 0 if all installed DIMMs are 88-bit single-node DIMMs. dcm returns a 1 if one or more 80-bit multinode DIMMs are detected.

When invoked with **`dcm -d all <node id>`**, dcm returns 0 and prints a table with the following format for a node with two EWMBs installed that were half-populated with 88-bit DIMMs:

Output table using `dcm -d all <node_id>`

```
Stingray Configuration Map DIMM Info:   Node:  0(hw2b-0000)
     ============================================================
    VERSION: 0.8.0.1 compiled: 1998/10/23 14:34:01
Memory Type:
============
      Physical: 88=Multi node 88-bit DIMM, 80=Single node 80-bit DIMM
 (Only physical DIMM type is reported.)
      * = Software Deconfigured     - = Not In Use
EWMB0:
======
EWMB0: Q0B0 88/88 Q1B4 88/88   Q2B0 -/-     Q3B4 -/-
EWMB0: Q0B1 88/88 Q1B5 88/88   Q2B1 -/-     Q3B5 -/-
EWMB0: Q0B2 88/88 Q1B6 88/88   Q2B2 -/-     Q3B6 -/-
EWMB0: Q0B3 88/88 Q1B7 88/88   Q2B3 -/-     Q3B7 -/-
EWMB1:
======
EWMB1: Q0B0 88/88 Q1B4 88/88   Q2B0 -/-     Q3B4 -/-
EWMB1: Q0B1 88/88 Q1B5 88/88   Q2B1 -/-     Q3B5 -/-
EWMB1: Q0B2 88/88 Q1B6 88/88   Q2B2 -/-     Q3B6 -/-
EWMB1: Q0B3 88/88 Q1B7 88/88   Q2B3 -/-     Q3B7 -/-
```

`dcm` returns a negative number for all scan-related failures.

dfdutil

`dfdutil` is a standalone offline utility that downloads firmware to SCSI devices including disks, arrays, and fibrechannel devices such as SCSI MUX and fibrechannel arrays.

The firmware image(s) are contained in a Logical Interchange Format (LIF) volume on the teststation at /spp/firmware/DFDUTIL.LIF. The raw (usually binary) firmware image of one or more devices is contained in the LIF filesystem. `dfdutil` reads this file when it initalizes and examines header of each file for a standard firmware header. The firmware header is required for download capability. Since most HP firmware distributions are already packaged in this format, the procedure for putting a raw binary firmware image into the proper format for `dfdutil` is not covered in this document.

NOTE DFDUTIL.LIF must have read permissions to be accessed by `dfdutil`.

To load and run `dfdutil`, enter the following command at the teststation prompt:

`tc_ioutil <node id> dfdutil.fw`

This command issues a system reset. The test controller bootstrap loads the executable image, dfdutil.fw, from the teststation file /spp/firmware/dfdutil.fw and executes it.

Once started, `dfdutil` loads the file DFDUTIL.LIF from the teststation and scans all SCSI and Fibrechannel busses on the system.

Example of dfdutil output when loading

```
Loading file dfdutil.fw
................................
........................................
.....................................................
.....................................................
..........................

dfdutil.fw copied successfully, booting


*****************************************************************************
***                              DFDUTIL                                  ***
***                              DFDUTIL                                  ***
***                                                                       ***
***              (C) Copyright Hewlett-Packard Co. 1998                   ***
***                        All Rights Reserved                            ***
***                                                                       ***
***  This program may only be used by HP support personneland             ***
***  those customers with the appropriate Class license or                ***
***  Node license for systems specified by the license.  HP               ***
***  shall not be liable for any damages resulting frommisuse             ***
***  or unauthorized use of this program.  This program                   ***
***  remains the property of HP.                                          ***
***                                                                       ***
***                    Version XXXXXXXXXXXXXXX                            ***
***                                                                       ***
*****************************************************************************

Please wait while I load the LIF file from the teststation.
Opening file /spp/firmware/DFDUTIL.LIF
transferring file. (Cursor will spin during Transfer)... |

        File name        Intended Product   ID            Rev.   Size
        ---------------- ---------------- -------------  ------ -----
        R18CUDA9         SEAGATE          ST19171W       0018  257888
        R23CUDA9         SEAGATE          ST19171W       0023  257888
        ST34371W84       SEAGATE          ST34371W       0484  276512
        GALAX03          DGC              C3400WDR5      0315  72864
        GALAX04          DGC              C3400WDR5      0415  72864
        GALX_IBM24       IBM              DCHS09F        2433  5872
        GALX_IBM53       IBM              DCHS09F        5333  5872
        ST118273         SEAGATE          ST118273WC     HP04  303360
        ST11827305       SEAGATE          ST118273WC     HP05  303360
        MUX1             HP               FC-SCSI_MUX    40_1  2162516
        MUX2             HP               FC-SCSI_MUX    d373  2162516
        DVD316           PIONEER          DVD303         0016  132594
        DVD317           PIONEER          DVD303         0016  132594
```

Example of dfdutil output (continued//0

```
 Indx   Path                    Product ID              Bus       Size      Rev
 ----   -------------------     -------------------     ----     ------     ------
  0     5/0.8.0.255.7.12.0      HP        HPA3308       FC        0         d373
  1     5/0.8.0.124.0.14.0      DGC       DISK          FCMUX     4006      0860
  1.0   ^array^                 SEAGATE   ST15150N      SCSI      4024      HP02
  2     5/0.8.0.124.1.5.0       HP        C5447A        FCMUX     0         HP06
  2.0   ^array^                 SEAGATE ST318275LC      SCSI      17366     HP00
  2.1   ^array^                 SEAGATE ST39102LC       SCSI      8683      HPE2
  2.2   ^array^                 SEAGATE ST39102LC       SCSI      8683      HPE2
  2.3   ^array^                 SEAGATE ST34572WC       SCSI      4095      HP03
  2.4   ^array^                 SEAGATE ST118202LC      SCSI      17366     HPE2
  2.5   ^array^                 SEAGATE ST118202LC      SCSI      17366     HPE2
  2.6   ^array^                 SEAGATE ST34572WC       SCSI      4095      HP03
|  3     4/2:0.3.0               SEAGATE ST19171W        SCSI      8683      0018
```

The output has a list of the valid firmware files in the LIF volume as well as a listing of all devices found in the scan of the SCSI/FC busses.

`dfdutil` bootable device table

The descriptions of the fields in the bootable device table are as follows:

- Index—Specifies in the `DOWNLOAD` command which device is used to download firmware to. FRUs in an array (the individual drives) are shown with a subindex (X.Y), where X is the array controller, and Y is the index of the physical drive. Array drives must be specified with the X.Y notation.
- Path—Specifiees the hardware path to the drive. There are two possibilities: fibrechannel or direct attach SCSI. In the case of directly attached SCSI, the path is formatted as `a/b:c.d.e`. Each letter in the path is defined as follows:
  - a—SAGA number
  - b—slot number
  - c—path level (always 0)
  - d—target ID
  - e—LUN number

  In the case of Fibrechannel bus, there are two possibilities: direct attach fibrechannel or fibrechannel SCSI MUX. The path of the direct attach fibrechannel is formatted as `a/b.c.d.255.e.f.g` with the definition of the letters as followes:
  - a—SAGA number

- b—slot number
- c—path level (always 0)
- d—always 8 for FC storage
- e—upper 4 bits of loop address
- f—lower 4 bits of loop address
- g—LUN number

If the device is attached to an FC MUX, the path is formatted as `a/b.c.d.e.f.g.h`. with the letter definitions as followes:

- a—SAGA number
- b—slot number
- c—path level (always 0)
- d—always 8 for FC storage
- e—loop address (fibrechannel loop address of the MUX to which this device is attached)
- f—backside SCSI bus number
- g—target number
- h—LUN number

NOTE Array drives (FRUs) are not listed with an absolute hardware path since they are not directly accessable from the SCSI bus. They are listed with the special token "^array^" in the path field.

- Product ID—Specifies the ID strings read from the device using the `INQUIRY` command.
- Bus Type—Specifies the bus type, whether SCSI, FC, or FCMUX. SCSI indicates this device is directly attached to a wide or LVD SCSI bus. FC indicates the device is directly attached to a fibrechannel bus. FCMUX indicates the device is attached to the backside WD SCSI bus of a SCSI MUX.
- Size—Specifies the size of the firmware file in bytes.
- Revision level—Specifies the revision reported by the device in the inquiry data.

dfdutil LIF file table

The descriptions of the fields in the LIF file table are as follows:

- Filename—Specifies the name of the file in the LIF volume. The operator specifies this name when issuing download commands to the devices.
- Intended Product ID—Specifies the vendor name and Device product name. These fields are setup when the raw firmware is packaged for distribution. It may or may not match exactly the product and vendor Ids reported by the device INQUIRY data. Do not download firmware unless you are sure it is the proper firmware for the device. When attempting to download firmware where this string doesn't match the inquiry data, user will be prompted for confirmation before downloading.
- Rev.—Specifies the firmware revision of the file. This is also setup during firmware packaging.
- Size—This is the size in bytes of the file not including the file header.

dfdutil commands

The `DFDUTIL>` prompt indicates that the built-in command line interpreter is waiting for a command.

The commands available to this command line interpreter are:

- `DOWNLOAD <filename> <index>`
- `DISPMAP <index>`
- `DISPDILES`
- `RESET`
- `LS`
- `HELP` [command]
- `UTILINFO`

DOWNLOAD command

Use the DOWNLOAD command to download firmware to a particular device. DOWNLOAD transfers the contents of a particular firmware file to a device. It prompts the user for any arguments that were not specified on the command line.

NOTE Once the download begins, do not interrupt the process, or the devices to which the firmware is being loaded could be rendered useless.

The syntax for the DOWNLOAD command is:

```
DISMAP <filename> <disk index>
```

filename must match one of the file names in the LIF file table, and index must match one of the index numbers in the bootable device list (displayed when the program starts). If the file specified does not have the same vendor and product ID as the device whose index number is specified, an error message will be issued to the operator and the download will be aborted.

As an example, to download firmware to the SCSI MUX HPA3308 (the mux controller firmware), enter the following command line:

```
DFDUTIL> download MUX1 0
```

dfdutil prompts the user for confirmation since FC-SCSI_MUX does not match the product ID of the device, HPA3308.

To download to FRUs in an array, enter the following command line:

```
DFDUTIL> download firmware_file_name 2.1
```

DISPMAP command

The DISPMAP command displays a list of all devices connected to the system. The information displayed includes:

- Index number
- Product identification
- Device size
- Index number
- Firmware revision

The syntax for the this command is:

`DISPMAP <disk index>`

The user may enter the index number of a single device; using no index number causes `DISPMAP` to list all devices.

This command will display the bootable device table displayed when `dfdutil` is started. If the optional argument [index] is specified, then only the information for the given index number will be displayed, not the entire table. This display may not reflect any downloads that may have been done since the program was started.

The following two examples show output using no index number and one index number, respectively.

Example output of `dfdutil DISPMAP` command with no index number

```
Indx Path         Product ID                  Bus    Size   Rev
---- ------------ ----------------------------- ------ ------ ----
0    2/0/1.2.0    QUANTUMLP270S disc drive      SCSI   258 MB 5909
1    2/0/1.6.0    QUANTUMLP270S disc drive      SCSI   258 MB 1234

Legend:
Indx = Index number used for referencing the device
Rev  = Firmware Revision of the device

Note:  Due to different calculation methods used, the size
       of the device shown is only a rough approximation.

DFDUTIL>
```

Example output of `dfdutil DISPMAP` command with one index number

```
Indx Path         Product ID                  Bus    Size   Rev
---- ------------ ----------------------------  ------ ------ ----
1    2/0/1.6.0    QUANTUMLP270S disc drive      SCSI   258 MB 1234

Legend:
Indx = Index number used for referencing the device
Rev  = Firmware Revision of the device

Note:  Due to different calculation methods used, the size
       of the device shown is only a rough approximation.

DFDUTIL>
```

DISPFILES command

The DISPFILES command displays a list of all available firmware files found on a LIF device. The command displays:

- File name
- Intended product identification
- New revision number
- Size of firmware (not file size)

The syntax for this command is:

```
DISPFILES
```

The user may enter the index number of a single device; using no index number causes DISPFILES to list all devices.

LS command

The LS command displays information about the LIF volume. The display is similar to that displayed by a `lifls -l` command. This command is used for writing and maintaining `dfdutil`.

RESET command

The RESET command only resets the internal variables of the `dfdutil` utility by resetting all variables and lists of original values. It rescans each bus to detect any devices. It does not reset any SCSI buses. Therefore, the resets display produced may not reflect any downloads that may have been done since `dfdutil` was started.

The syntax for this command is:

```
RESET
```

UTILINFO command

This command provides general information about the use of `dfdutil`.

HELP command

The HELP command provides useful information about `dfdutil` commands.

The syntax for this command is:

```
HELP <command name>
```

Entering **HELP** without a command name displays a list of all available `dfdutil` commands. Entering the specific command name after **HELP** outputs specific information about the command.

Notes and cautions about `dfdutil`

This section presents some limitations and cautions concerning `dfdutil`.

Backup before downloads

Some firmware downloads may affect formatting resulting in the loss of some or all the data on the disk.

CAUTION Back up all disks before loading firmware onto them.

Halting the system during downloads

Halting the system during a download may leave the drive being downloaded in an unusable state.

CAUTION Never halt the computer, power cycle it, or in any way interrupt operation during a download.

Power cycling after a download

Some disk drives store downloaded code to nonvolatile memory but do not load and run this code until after the next bus reset or power cycle.

NOTE Power cycle the system and all cabinets or racks containing drives that have been downloaded after all downloads have been completed. Restart `dfdutil` and examine the revision levels in the bootable device table to make sure that all downloads were successful.

If attempting to download a corrupted or inappropriate firmware file to some drives, the drives drop the downloaded data and return good status. For this reason, `dfdutil` can not always determine if a download did, in fact, complete successfully.

Shared SCSI Buses

If `dfdutil` is running on a system which shares any of its SCSI busses with another system or systems, the other system or systems must be halted while this program is running. This program can not determine that a bus is shared, so the operator must determine if any bus is shared and halt the other computer(s).

Shared Nike Arrays

If `dfdutil` is running on a system which shares a Nike array with another system, it is not possible to update firmware on the Nike's SP boards or drives without manual intervention. This program can detect that the array is shared and display a message to pull the SP board connected to the other system and reinsert the board after the download(s) is complete.

Nike and Galaxy drive download to the individual disks in the array is not possible with two active SP controllers in the cabinet. One SP must either be physically removed or shut down via remote maintenance software (accessed via the serial port).

dump_rdrs

The `dump_rdrs` utility automatically resets the specified node and directs it to boot the RDR dumper firmware module. Once it detects that the RDR dumper firmware has completed, it scans out the results and places a formatted RDR dump of each processor in /spp/data/<complex>/nodeX.cpuY.rdrs. X is the node number specified and Y is a processor number from 0 - 31.

Example of dump_rdrs utility:

```
dump_rdrs <node id>
```

fwcp

fwcp is an OBP command that upgrades system firmware. A single firmware package may be loaded by the following command:

```
% fwcp <filename>
```

To load all system firmware packages, use the following master download script:

source /core@f0,f0000000/ lan@0,d30000;15.99.111.99:/spp/scripts/dl-diags

The master download script output is shown below:

```
v-c-t:/spp/firmware$ cat /spp/scripts/dl-diags
fwcp 15.99.111.99:/spp/firmware/pdcfl.fw PDCFL
fwcp 15.99.111.99:/spp/firmware/post.fw POST
fwcp 15.99.111.99:/spp/firmware/test_controller.fw TC
fwcp 15.99.111.99:/spp/firmware/cpu3000.fw CPU3000
fwcp 15.99.111.99:/spp/firmware/io3000.fw IO3000
fwcp 15.99.111.99:/spp/firmware/mem3000.fw MEM3000
fwcp 15.99.111.99:/spp/firmware/diodc.fw DIODC
fwcp 15.99.111.99:/spp/unsupported/rdr_dumper.fw
RDR_DUMPER
fwcp 15.99.111.99:/spp/firmware/entry2500.pdc /flash@0,0
fwcp 15.99.111.99:/spp/firmware/obp2500.pdc OBP
```

fw_init

fw_init provides an automatic means for downloading firmware to each node and initializing certain data structures in NVRAM. Using this script prevent problems that could occur when executing this procedure manually. The format if fw_init is as follows:

```
fw_init [-c complex name]
```

-c complex name specifies the complex to update.

For example

fw_init	updates all nodes in the current complex.
fw_init -c hw2a	updates all nodes in the complex hw2a.

If the -c option is not specified, then the complex_name value is obtained either from an environment variable of the same name or it defaults to mu.

fw_init first loads the JTAG core firmware, JTAG firmware, and the diagnostic LIF header to each node in the complex. The complex is then reset to OBP in order to download firmware to all the nodes. A source command is issued to OBP that loads all the firmware listed in the /spp/scripts/dl-diags file into flash memory. After this completes, the tc_init utility is executed which initializes certain NVRAM data structures used by the Test Controller.

This script must be executed as root. If not then an error message is printed and the script terminates. The error message is as follows:

```
This script must be run as root.
```

Messages are periodically printed to the console while fw_init is executing. Examples of these messages are show below:

fw_init message example 1

```
Starting the firmware download and initialization
process.
```

fw_init message example 2

```
Loading JTAG core firmware on "hw2a-0000".
```

fw_init message example 3

```
Loading Diagnostic LIF header on "hw2a-0000".
```

fw_init message example 4

```
Loading JTAG firmware on "hw2a-0000".
```

fw_init message example 5

```
The "hw2a" complex will now be reset to OBP.  Please wait
```

fw_init message example 6

```
Saving NVRAM contents and beginning firmware download
via OBP.
```

fw_init message example 7

```
Now clearing NVRAM and resetting the system again.
Please wait.
```

fw_init message example 8

```
Now restoring NVRAM.  Please wait.
```

fw_init message example 9

```
Initializing the test_controller data structures.
```

fw_init message example 10

```
Performing any file cleanup and miscellaneous tasks.
```

fw_init message example 11

```
The firmware download and initialization has been
completed.
```

get_node_info

The get_node_info utility provides as a mechanism for scripts or programs to access the teststation configuration information generated by the ts_config configuration tool. It has the following format:

```
get_node_info [node_info] [OPTIONS]
```

When a V2500 node is configured by ts_config, an entry is added to a node configuration file. Each node entry contains the following information:

- Complex Name—Complex name assigned in ts_config
- Node ID—V Class Node ID
- Diagnostic IP hostname—IP hostname of Diagnostic Utility interface
- OBP IP hostname—IP hostname assigned to OBP LAN interface
- Teststation Diagnostic hostname—IP hostname assigned to TS Diag interface
- Console name—Name assigned to V Class console

get_node_info obtains the teststation configuration information about all nodes or a single node. If -A is used to request information on all nodes, the node entries are returned in the order they appear in the configuration file (they are not sorted).

By default, the information returned includes all of the configuration fields. OPTIONS select a subset of the available fields. The output fields are returned (to standard output) in the order shown above, regardless of the ordering of OPTIONS.

[node_info] must uniquely identify a node on the teststation, a Node ID (for example, 0) or the Diagnostic IP hostname (for example, swtest-0000)

If a Node ID is specified, get_node_info determines the node Complex Name from the COMPLEX_NAME environment variable. Use the set_complex command to set the desired complex name.

[OPTIONS] include the following:

- -a—Display all fields (default)
- -A—Display all configured nodes

The selected fields will be printed in the order below)

- -c—Display the Complex name
- -n—Display the Node id
- -m—Display the Diagnostic IP hostname
- -o—Display the OBP IP hostname
- -t—Display the Test Station Diagnostic hostname
- -s—Display the console name

The following are examples of the get_node_info utility:

Example showing the return all information about Node Id 0:

```
joker-t(hw2a):/users/sppuser$ get_node_info 0
hw2a 0 hw2a-0000 obp-hw2a-0000 tsdart-d Serial_1 2
```

Example of retuening the complex name associated with the Diagnostic name

```
joker-t (hw2a): /users/sppuser$ get_node_info hw2a-0000 -c
hw2a
```

The sppconsole script contains an example use of the get_node_info utility.

hard_logger

hard_logger is a script that invokes the interrogators and extractors to log all error information on a node

The usage of the script is:

```
hard_logger [node number]
```

[node number] is a hex number.

hard_logger resides in /spp/scripts/hard_logger and is automatically invoked be ccmd when a hard error occurs.

The hard_logger script performs the following tasks:

- Parses the command line arguments to determine on which node it should run. ccmd sets up the COMPLEX_NAME environment variable before invoking hard_logger. The teststation utilities called by hard_logger use the combination of node_id and COMPLEX_NAME to determine with which node to communicate.
- Acquires COP information for the node using sppdsh and saves the output to /spp/data/<COMPLEX_NAME>/hl/T_FILE_n$node
- Acquires PCE information using sppdsh and saves the output to /spp/data/<COMPLEX_NAME>/hl/T_FILE_n$node.
- Checks the Stop On Hard bits of each SPAC to find one that is running. If an SPAC is running, then hard_logger gets information from the SMUC CSRs.
- Reads SMUC CSRs. If there is no hard error, hard_logger quits.
- Traverses the list of hard error buses. If a bus reports a hard error, then it performs the following:
  - Interrogates each controller on that bus for hard errors.

    If the hard error group pin is set to a one value, it ignores the controller.

    If the pin is a zero value, the controller may have been the first to record the error.
  - Interrogates the controller reporting the error.

To interrogate the controllers, `hard_logger` calls the ASIC specific interrogator located in /spp/scripts/<asic>.

For example, the SMAC interrogator is located in /spp/scripts/smac

The interrogator returns a list of extractors to run on that ASIC in /spp/data/<COMPLEX_NAME>/hl/inter_n$node.

- Runs each extractor returned by the interrogator.

- Sends the COP, PCE, interrogator, and extractor output to `event_logger`. `event_logger` forwards the COP, PCE, and extractor output to both the teststation message console window and the `ccmd` log file /spp/data/ccmd_log.
- Logs the results in /spp/data/<complex_name>/hard_hist.

  hard_hist is a permanent file that records the date, time, and results of the script.

The last run of `hardlogger` on a node is preserved in /spp/data/<complex_name>/hl/OUTPUT_FILE_n$node.

lcd

`lcd` prints the current contents of the liquid crystal display for node 0 of the current complex. It has the following format:

```
lcd
```

The complex can be changed by using the `set_complex` utility. The output is sent to `stdout` output.

Example output of `lcd`

```
0 (0,0)
I-I- ---- I-P- ----
---- ---- ---- ----
abcedfghijklr-
```

load_eprom

The `load_eprom` utility resides on the teststation. It downloads the core firmware products into the EEPROM on the Utilities board through the scan interface. It can also update the JTAG scan interface controller firmware. If, during a download, it detects any errors, it automatically retries the download.

The `load_eprom` utility uses subroutines that perform the following functions:

- It reads a raw binary file on the teststation.
- It erases the specified Flash sector and verifies that the erase was successful. It will retry if the erase fails.
- It scan downloads the contents of the binary in 4096-byte page increments, updating the screen for each page. A “w” is printed during the write operation, an “r” during the optional read operation, a “v” during the optional verify operation and a “.” when the page is complete.
- It can optionally read each page back for verification.
- It can read-verify a binary in the Flash EEPROM and compare it to the binary on the teststation, without performing the write operation.

The `load_eprom` utility usage is as follows:

```
load_eprom -n <IP name> [-QRV] [-P #] [-j|c|e|p|o|t|l|f]
<file>
```

The options available are given in Table 83.

Table 83 **`load_eprom` options**

Option	Description
`-Q`	Quiet (no) output mode.
`-R`	Read and verify data only-No writing.
`-P number`	SPAC to use for scan operations where number is 0-7, 8 is UBUS.
`-V`	Verify data after a write.
-j <file>	Load binary into JTAG flash.
`-c <file>`	Load binary into JTAG_CORE flash.
`-e <file>`	Load binary into PDC Entry section.
`-p <file>`	Load binary into PDC POST section.
`-o <file>`	Load binary into PDC OBP section.
`-t <file>`	Load binary into PDC Test Controller section.
`-l <file>`	Load binary into LIF file section.
`-f <file>`	load binary into PDC firmware loader section

As an example, entering the following reads the file /spp/firmware/post.fw and updates the POST section of Flash EEPROM on the Utilities board.

```
xns3_d% load_eprom -n hw2a-0000 -p /spp/firmware/post.fw
```

Entering the following reads the file ./jtag.fw and updates the Flash EEPROM for the JTAG controller:

```
xns3_d% load_eprom -n hw2a-0000 -j jtag.fw
```

The following are three addition examples of the `load_eprom` command. The first two write to one sector in EEPROM and the last writes across several sectors.

Example output of `load_eprom -n hw2a-0000 -p entry.pdc` command

```
Reading file “entry.pdc”: 4253 (0x109d) bytes read.
Using default SPAC (P0L).
Erasing sector 0  (0xf0000000) OK
Writing sector 0  (0xf0000000) .. OK
```

Example output of `load_eprom -n hw2a-0000 -p post.fw` command

```
Reading file “post.fw”: 92820 (0x16a94) bytes read.
Using default SPAC (P0L).
Erasing sector 4  (0xf0020000) OK
Writing sector 4  (0xf0020000) ..................... OK
```

Example output of `load_eprom -n hw2a-0000 -o obp.pdc` command

```
Reading file “obp.pdc”: 499712 (0x7a000) bytes read.
Using default SPAC (P0L).
Erasing sector 7  (0xf0080000) OK
Writing sector 7  (0xf0080000) ............................. OK
Erasing sector 8  (0xf00a0000) OK
Writing sector 8  (0xf00a0000) ............................. OK
Erasing sector 9  (0xf00c0000) OK
Writing sector 9  (0xf00c0000) ............................. OK
Erasing sector 10 (0xf00e0000) OK
Writing sector 10 (0xf00e0000) ......................... OK
```

While `load_eprom` is writing a block of data, a “w” is printed. If the write is successful, a dot is printed. The dots continue until the whole sector is successfully written, at which time the “OK” is printed.

pim_dumper

pim_dumper is a utility used to display Process Internal Memory (PIM) information after a TOC, LPMC, or HPMC. The PIM dump information includes the processor registers and various ASIC registers. It has the following fomat:

```
pim_dumper [-c CPU#] [-n NODE_PARM] [-t][-l][-h] [-e][-help]
```

Example of pim_dumper use:

pim_dumper -h -c 2

This example displays HPMC information for Processor 2 on Node 0.

The PIM information will be appended to the file /spp/data/<COMPLEX_NAME>/pimlog, where <COMPLEX_NAME> is the name associated with the desired node. Optionally, a copy of the PIM information can be written to standard output.

pim_dumper can be invoked without any command line options. By default, it dumps all available (TOC/LPMC/HPMC) information for all enabled processors on node 0.

Table 84 lists pim_dumper options.

Table 84 pim_dumper options

Option	Description
-c CPU number	Request a specific processor
-c all	Select all processors (default: all)
-n NODE_PARM	Specify the desired node ID (default: 0) or node name (e.g. test-0000)
-t	Display TOC information (default: on)
-l	Display LPMC informationa (default: on)
-h	Display HPMC information (default: on)
-e	Echo PIM to standard output (default: off)
-help	Display usage information

The TOC/LPMC/HPMC options are mutually exclusive. Specify only one of these options; do not specify any, and the default mode dumps all TOC/LPMC/HPMC data.

If `pim_dumper` is able to accomplish the desired action, it returns zero . If for any reason the requested operation cannot be completed, a non-zero exit code is used.

set_complex

The `set_complex` sets the default V2500 Complex Name in the current shell environment.

```
set_complex [COMPLEX_NAME]
```

Once set, teststation diagnostic or console utilities that are run from within the shell operate on the specified complex.

If multiple complexes are configured on a single teststation, individual shells can each be set to a specific default complex using `set_complex`. Diagnostic and console commands entered from the shell access the desired node as if it were the only complex on the teststation.

Example of command entered from the shell

```
joker-t (hw2a): /users/sppuser$ sppconsole 0
```

In this example, the command accesses the console for Node ID 0 in the hw2a complex.

Users may temporarily override the default complex by including the full Diagnostic Node name in the Diagnostic or console command. For example, even though the default complex is set to hw2a, the following command requests `flash_info` from Node ID 0 in the hw2b complex:

```
joker-t(hw2a):/users/sppuser$flash_info hw2b-0000
```

NOTE

`jf-ccmd_info` lists the diagnostic node names for all active nodes on the teststation Diagnostic LAN.

The customized shell environment for the sppuser account automatically runs `set_complex` during login. If a single V2500 complex has been configured, the default `COMPLEX_NAME` is assigned automatically. If more than one complex is configured, the user is prompted for the desired complex. With help from the parent shell, `set_complex` causes the `COMPLEX_NAME` environment variable to be set appropriately.

`set_complex` also updates the shell prompt to reflect the default complex name. The complex name is enclosed in parenthesis in the prompt string. If the shell is running on the teststation desktop, `set_complex` also updates the shell window title.

`set_complex` can be invoked anytime the user wants to change the shell default complex. If the user enters an invalid `COMPLEX_NAME`, the default complex becomes unset and the prompt string indicates this condition. If the user does not enter a `COMPLEX_NAME`, the complex name remains set (assuming it is still a valid complex).

`set_complex` does not work from within a shell script. An alternative is to explicitly set the COMPLEX_NAME environment variable using the appropriate mechanism for the current shell script type.

Example showing change of complex name in a shelll script

```
++++++++++++++++++++++++++++++++++++++++++++
+#!<shell>                                  +
+                                           +
+ COMPLEX_NAME=hw2a; export COMPLEX_NAME    +  (sh/ksh/sppdsh)
+ <OR>                                      +
+ setenv COMPLEX_NAME hw2a                  +  (csh/tcsh)
+                                           +
+ dcm 0                                     +
+ ...                                       +
++++++++++++++++++++++++++++++++++++++++++++
```

NOTE Scripts that are run from a shell using `set_complex` receive the correct COMPLEX_NAME environment variable from the parent shell. The limitation is that `set_complex` cannot set the `COMPLEX_NAME` environment variable when run from within a script.

soft_decode

`soft_decode` decodes single-bit ECC error data. This `perl` script decodes single-bit ECC error information. It prompts for syndrome, row, and address information that is parsed, decoded, and displayed in an easy-to-read format that can be cut-and-pasted into quasar.

To exit enter **q**.

Example of `soft_decode` use:

```
% soft_decode

Enter RAM size (16, 64 or 128): 16
Enter syndrome code: 64
Enter row number: 2

Enter address: 04589030
Single-Bit ECC Error RAM Information
====================================
Location      Bit    Pin#    Row   Address
--------      ---    ----    ---   -------
U013A7       DQ2      6      2     589030
Enter RAM size (16 or 64): q
exiting
```

sppconsole

`sppconsole` connects the user to the console for a specified node.

`sppconsole` has the following format:

```
% sppconsole node [opt1, ..., optN
```

There are several ways to initiate the `sppconsole` interface.

- Run the `sppconsole` command in a shell on the teststation.
- Select from the teststation root menu the desired V2500 complex, then select “Console” and the desired node.
- Use the `consolebar` utility to select the desired node.

The `sppconsole` script invokes the /spp/etc/console program (passing any optional arguments and the node number) to provide the console interface to the V2500 node. This interface communicates with POST, OBP, the Test Controller, and the HP-UX operating system. It starts up a window and connects the user to the console server, that is the `conserver` daemon, running on the test station. After making the connection, the last 20 lines of the console output are displayed.

The `conserver` daemon is started by `init` when the teststation is booted. The daemon reads the /spp/data/conserver.cf file to determine which console terminals to open and maintain.

All errors and information messages are logged in the system log file /var/adm/syslog/syslog.log.

The `sppconsole` script invokes the `/spp/etc/console` program to provide the operating system a console interface. Refer to the `console(8)` man page for more information about this program.

The following shows the typical output in the console window when the node boots.

Example of `sppconsole` boot output

```
joker-t(hw2b)% sppconsole
[enter `^Ec?' for help]
[no, sppuser@joker-t is attached]
[replay]
POST Hard Boot on [0:PB0L_A]
HP9000/V2500 POST Revision 1.0.0.1, compiled 1998/12/03 09:50:10 (#0039)
Probing CPUs: PB0L_A PB1R_A PB2L_A PB3R_A PB4L_A PB5R_A PB6L_A PB7R_A
Completing core logic SRAM initialization.
Starting main memory initialization.
 Probing memory: MB0L MB1L MB2R MB3R MB4L MB5L MB6R MB7R
 Installed memory: 24576 MBs, available memory: 13312 MBs.
 Initializing main memory.
  Parallel memory initialization in progress.
                  r0         r1         r2         r3
  PB0L_A MB0L [:::: ::::][:::: ::::][:::: ....][:::: ....]
  PB1R_A MB1L [:::: ::::][:::: ::::][:::: ....][:::: ....]
  PB2L_A MB2R [:::: ::::][:::: ::::][:::: ....][:::: ....]
  PB3R_A MB3R [:::: ::::][:::: ::::][:::: ....][:::: ....]
  PB4L_A MB4L [:::: ::::][:::: ::::][:::: ....][:::: ....]
  PB5R_A MB5L [:::: ::::][:::: ::::][:::: ....][:::: ....]
  PB6L_A MB6R [:::: ::::][:::: ::::][:::: ....][:::: ....]
  PB7R_A MB7R [:::: ::::][:::: ::::][:::: ....][:::: ....]
 Building main memory map.
Main memory initialization complete.
Booting OBP
```

After POST initializes the system, OBP boots. The following is a sample of the output.

Example of OBP output while booting

```
OBP Power-On Boot on [0:0]
--------------------------------------------------------------------------------
                       PDC Firmware Version Information
                            PDC_ENTRY version 4.1.0.9
                             POST Revision: 1.0.0.1
          OBP Fieldtest Release 4.1.0.9, compiled 98/10/30 14:11:20  (3)
                 SPP_PDC Fieldtest 1.4.0.19 (11/12/98 19:17:49)

--------------------------------------------------------------------------------
Proc type   Proc#   Proc Rev  Speed     State   Dcache   Icache   I-prefetch
----------  -----   --------  -------  -------  -------  -------  ----------
HP,PA8500   0       2.0       440 MHz  Active   1024 KB  512 KB   On
HP,PA8500   2       2.0       440 MHz  Active   1024 KB  512 KB   On
HP,PA8500   4       2.0       440 MHz  Active   1024 KB  512 KB   On
HP,PA8500   6       2.0       440 MHz  Active   1024 KB  512 KB   On
HP,PA8500   8       2.0       440 MHz  Active   1024 KB  512 KB   On
HP,PA8500   10      2.0       440 MHz  Active   1024 KB  512 KB   On
HP,PA8500   12      2.0       440 MHz  Active   1024 KB  512 KB   On
HP,PA8500   14      2.0       440 MHz  Active   1024 KB  512 KB   On
Primary boot path   = 0/0/0.6.0
Alternate boot path = 15/3 NFS 15.99.111.99:/spp/os/uxinstlf
Console path        = 15/1
Keyboard path       = 15/1
[*** Manufacturing (or Debug) Permissions ON ***]
System is HP9000/800/V2500 series
Autoboot and Autosearch flags are both OFF or we are in HP core mode.
Processor is entering manual boot mode.
Command                          Description
-------                          -----------
AUto [BOot|SEArch ON|OFF]        Display or set the specified flag
BOot [PRI|ALT|<path> <args>]     Boot from a specified path
BootTimer [time]                 Display or set boot delay time
CLEARPIM                         Clear PIM storage
CPUconfig [<cpu>] [ON|OFF|SHOW] (De)Configure/Show Processor
DEfault                          Set the system to defined values
DEfault                          Set the system to defined values
DIsplay                          Display this menu
ForthMode                        Switch to the Forth OBP interface
IO                               List the I/O devices in the system
LS [<path>|flash]                List the boot or flash volume
OS [hpux|sppux]                  Display/Select Operating System
PASSword                         Set the Forth password
PAth [PRI|ALT|CON] [<path>]      Display or modify a path
PDT [CLEAR|DEBUG]                Display/clear Non-Volatile PDT state
PIM_info [cpu#] [HPMC|TOC|LPMC] Display PIM of current or any CPU
RESET [hard|debug]               Force a reset of the system
RESTrict [ON|OFF]                Display/Select restricted access to Forth
SCSI [INIT|RATE] [bus slot val] List/Set SCSI controller parms
SEArch [<path>]                  Search for boot devices
SECure [ON|OFF]                  Display or set secure boot mode
TIme [cn:yr:mo:dy:hr:mn[:ss]]    Display or set the real-time clock
VErsion                          Display the firmware versions
Command:
```

The following message appears in the console window:

```
[0:1] ok [read-only -- use `^Ecf' to attach, `^Ec?' for
help]
```

Attach to the node by entering **Ctrl** **ecf**.

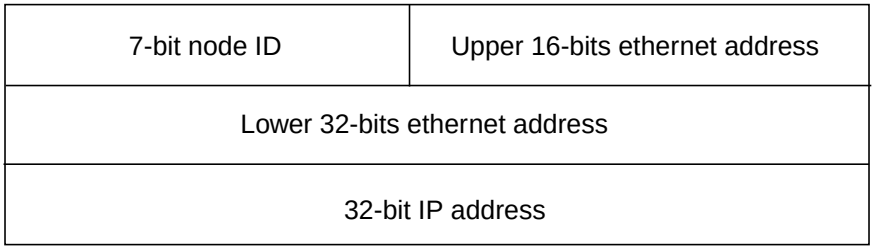
Press the **Ctrl** key **e** simultaneously; do not press the **Ctrl** key with the **c** and **f**.

All information and error messages are logged into the /usr/adm/syslog system error log file.

tc_init

`tc_init` determines the node ID, ethernet address, and IP address for all nodes in the complex. This information is then stored in the NVRAM of all nodes as one 12-byte entry per node. Each 12-byte entry has the format shown in Figure 54:

Figure 54 **`tc_init` NVRAM entry**



In addition, `tc_init` updates the ARP entries on the teststation by executing as root. If it can not execute as root, then the following error is displayed:

```
** This utility must be executed as root.
** Please login as root and try again.
```

`tc_init` outputs node information shown in the following example.

`tc_init` samle output

```
ex-c2-t% tc_init

  Node = 0 [index=0]
    7-bit node id    = 00
    Host Name        = obp-hw2a-0000
    Upper ether addr = 0x000000a0
    Lower ether addr = 0xd900adb3
    IP addr          = 0x0f636fa6 [15.99.111.166]
    ARP delete command = arp -d obp-hw2a-0000
  obp-hw2a-0000 (15.99.111.166) deleted
    ARP add command = arp -s obp-0000 0:a0:d9:0:ad:b3
```

Execute `tc_init` after the node has been configured by `jf-node_ip_set` and `xconfig`. `ccmd` must finish the scan database generation. Once `ccmd` executes, the changes become effective the next time `test_controller` is running. If `ccmd` is running when `tc_init` is executed then `test_controller` must be restarted.

`tc_init` only needs to be executed once. The following are the only reasons for having to rerun this utility:

- NVRAM is corrupted.
- The system is reconfigured

NOTE `tc_init` is run as part of the `fw_init` script and should not be run under normal circumstances.

tc_ioutil

`tc_ioutil` resets the node and requests that the Test Controller load, (via tftp) and boot the specified file. It has the following format:

`tc_iotuil <node id>  <file>`

`node id` may be a node number of ip name and `file` should be the name of a file in /spp/firmware

tc_show_struct

The `tc_show_struct` tool examines certain structures that the test controller uses to set up and run tests. It has the following format:

```
tc_show_struct <test_name> <node_number OR node_name>
```

Possible selections for the tests are:

- `-mem`
- `-io`
- `-cpu`

The structures examined are:

- `tc_global_parameter_struct`
- `tc_test_info_struct`
- `tc_cpu_info_struct`

The `tc_test_info_struct` structure displays the following fields:

- Entry point
- Class pointer
- Subtest pointer
- Hardware requirements
- Test specifics
- Parameter pointer
- HW requirements met
- Test initiated
- Test selected
- Run classes
- Run subtests

The `tc_global_parameter_struct` structure displays the master state.

The `tc_cpu_info_struct` structure displays the status or state of each processor and the current subtest.

The `tc_show_struct` tool takes two arguments: the first is the test of interest, the second is the node of interest.

Example of `tc_show_struct` output

```
joker-t(hw2b):/users/sppuser$ tc_show_struct -mem 0
------------------------------------------------------------------------
NODE 0  (hw2b-0000)
------------------------------------------------------------------------
Name : MEM3000 - EEPROM based memory tests
Entry Pt     ClTb ptr    StTb ptr      HwReq     ParmTbptr  Parm_ptr
------------------------------------------------------------------------
0xf01d0000  0xf01d0074  0xf01d02a8  0xf01d006c  0xf0840760  0xf0863158
      ------------------------------------------------------------------------
Hardware req met = 0 | Test inited = 1
Selected = 0         | TC State = TC_RUNNING
------------------------------------------------------------------------
Test error cnt = 0   | Loop enable = 0
Loop count = 0       | Paused mask = 0x00
------------------------------------------------------------------------
Class[0]   =          0    Subtest[0] =        640
Class[1]   =  538968128    Subtest[1] =          0
Class[2]   =    1050624    Subtest[2] =    8389636
Class[3]   =  537395328    Subtest[3] =   12588160
Class[4]   =         32    Subtest[4] =  537395200
------------------------------------------------------------------------
Current Values for Parameters
 00) 0xa5a5a5a5 01) 0xa5a5a5a5 02) 0x5a5a5a5a 03) 0x5a5a5a5a
 04) 0x00000007 05) 0x00000001 06) 0x00000002 07) 0x00000000
 08) 0x00000000 09) 0x00000000 10) 0x000000f0 11) 0x00000000
 12) 0x00000000 13) 0x00000000 14) 0x00000000 15) 0x00000000
 16) 0x00000000 17) 0x00000000 18) 0x00000000 19) 0x00000000
 20) 0x00000000 21) 0x00000000 22) 0x00000000 23) 0x00000000
 28) 0x00000000 29) 0x00000000 30) 0x00000000 31) 0x00000000
 32) 0x00000000 33) 0x00000000 34) 0x00000000 35) 0x00000000
 36) 0x00000000 37) 0x00000000 38) 0x00000000 39) 0x00000000
 40) 0x00000000 41) 0x00000000 42) 0x00000000 43) 0x00000000
 44) 0x00000000 45) 0x00000000 46) 0x00000000 47) 0x00000000
 48) 0x00000000 49) 0x00000000 50) 0x00000000 51) 0x00000000
 52) 0x00000000 53) 0x00000000 54) 0x00000000 55) 0x00000000
 56) 0x00000000 57) 0x00000000 58) 0x00000000 59) 0x00000000
 60) 0x00000000 61) 0x00000000 62) 0x00000000 63) 0x00000000
 64) 0x00000000 65) 0x00000000 66) 0x00000000 67) 0x00000000
 68) 0x00000000 69) 0x00000000 70) 0x00000000 71) 0x00000000
 72) 0x00000000 73) 0x00000000 74) 0x00000000 75) 0x00000000
 76) 0x00000000 77) 0x00000000 78) 0x00000000 79) 0x00000000
 80) 0x00000000 81) 0x00000000 82) 0x00000000 83) 0x00000000
 84) 0x00000000 85) 0x00000000 86) 0x00000000 87) 0x00000000
 88) 0x00000000 89) 0x00000000 90) 0x00000000 91) 0x00000000
 92) 0x00000000 93) 0x00000000 94) 0x00000000 95) 0x00000000
 96) 0x00000000 97) 0x00000000 98) 0x00000000 99) 0x00000000
100) 0x00000000 101) 0x00000000 102) 0x00000000 103) 0x00000000
```

```
104) 0x00000000 105) 0x00000000 106) 0x00000000 107) 0x00000000
108) 0x00000000 109) 0x00000000 110) 0x00000000 111) 0x00000000
112) 0x00000000 113) 0x00000000 114) 0x00000000 115) 0x00000000
116) 0x00000000 117) 0x00000000 118) 0x00000000 119) 0x00000000
120) 0x00000000 121) 0x00000000 122) 0x00000000 123) 0x00000000
124) 0x00000000 125) 0x00000000 126) 0x00000000 127) 0x00000000
------------------------------------------------------------------------
CPU Mask  = 0x0000    SPAC Mask = 0x00
SMAC Mask = 0x00      STAC Mask = 0x00
SAGA Mask = 0x00
------------------------------------------------------------------------
CPU 0  - State - TC_CPU_RUNNING                Subtest 310
CPU 1  - State - TC_CPU_RUNNING                Subtest 310
CPU 2  - State - TC_CPU_RUNNING                Subtest 310
CPU 3  - State - TC_CPU_RUNNING                Subtest 310
CPU 4  - State - TC_CPU_RUNNING                Subtest 310
CPU 5  - State - TC_CPU_RUNNING                Subtest 310
CPU 6  - State - TC_CPU_RUNNING                Subtest 310
CPU 7  - State - TC_CPU_RUNNING                Subtest 310
CPU 8  - State - TC_CPU_RUNNING                Subtest 310
CPU 9  - State - TC_CPU_RUNNING                Subtest 310
CPU 10 - State - TC_CPU_RUNNING                Subtest 310
CPU 11 - State - TC_CPU_RUNNING                Subtest 310
CPU 12 - State - TC_CPU_RUNNING                Subtest 310
CPU 13 - State - TC_CPU_RUNNING                Subtest 310
CPU 14 - State - TC_CPU_RUNNING                Subtest 310
CPU 15 - State - TC_CPU_RUNNING                Subtest 310
CPU 16 - State - TC_CPU_NOT_AVAIL              Subtest 0
CPU 17 - State - TC_CPU_NOT_AVAIL              Subtest 0
CPU 18 - State - TC_CPU_NOT_AVAIL              Subtest 0
CPU 19 - State - TC_CPU_RUNNING                Subtest 310
CPU 20 - State - TC_CPU_NOT_AVAIL              Subtest 0
CPU 21 - State - TC_CPU_NOT_AVAIL              Subtest 0
CPU 22 - State - TC_CPU_NOT_AVAIL              Subtest 0
CPU 23 - State - TC_CPU_NOT_AVAIL              Subtest 0
CPU 24 - State - TC_CPU_NOT_AVAIL              Subtest 0
CPU 25 - State - TC_CPU_NOT_AVAIL              Subtest 0
CPU 26 - State - TC_CPU_NOT_AVAIL              Subtest 0
CPU 27 - State - TC_CPU_RUNNING                Subtest 310
CPU 28 - State - TC_CPU_RUNNING                Subtest 310
CPU 29 - State - TC_CPU_NOT_AVAIL              Subtest 0
CPU 30 - State - TC_CPU_NOT_AVAIL              Subtest 0
CPU 31 - State - TC_CPU_RUNNING                Subtest 310
```

Version utilities

This section describes the three version utilities.

diag_version

The `diag_version` utility displays the product name and the version of the current teststation software. For example:

```
$ diag_version
HP9000/V2500 Diagnostics, Version 1.0.0.0
```

flash_info

`flash_info` reads the known entry points for the various products that are stored in flash EEPROM. If they have the correct magic number and the pointer to the version string is not null, the version string is extracted.

If no argument is provided, the lowest node in the complex is used. The node number is entered in hexadecimal.

Example of `flash_info` output

```
joker-t (hw2a):/users/sppuser$ flash_info 0

Node : 0 (hw2a-0000)

  Program Name       Version            Date        Build Level
-------------------------------------------------------------------
pdcfl                1.0.0.0          1998/11/16        0006
post                 1.0.0.0          1998/10/26        0114
dump_rdrs            1.0.0.0          1998/12/14
test_controller      1.0.0.0          1998/10/07        0002
mem3000              1.0.0.0          1998/10/26        0423
cpu3000              1.0.0.0          1998/10/01        0001
io3000               1.0.0.0          1998/10/23        0043
diodc                1.0.0.0          1998/10/07        0004
obp                  4.0.0.8
pdc_entry            4.0.0.8
```

ver

ver is a teststation version retriever utility. It is used to read and display the version information built into each diagnostic product. Its usage is:

ver <file>

ver searches the specified file for a version string previously compiled or inserted into the file and extracts and displays a version and date stamp. This works for most teststation utilities and diagnostics firmware. Special options are required to display OBP, Entry PDC, SPP PDC and Symbios Fcode firmware revisions, as shown below:

ver -e <Entry PDC file>

ver -o <OBP2500 file>

ver -p <SPP PDC file>

ver -s <Symbios Fcode file>

Event processing

This section discusses three event processing utilities:

- `event_logger`
- `log_event`

event_logger

The `event_logger` utility is the teststation Event Logger and has a format as follows:

```
event_logger [-d]
```

`event_logger` receives messages from diagnostic utilities through `rpc` calls and writes them to the event log for later review or processing.

The `-d` option keeps `event_logger` from running as a daemon which is useful for debugging.

`event_logger` is a background daemon and is started by `init` through `inittab`. `event_logger` receives messages for the `event_log` via two different mechanisms. Teststation utilities programs send events to the `event_logger` through `rpc` calls. OS events on the other hand, use `UDP` datagrams that are sent out over a specific port. These must be detected and logged as well. Upon receiving an OS event, the event travels the same path as a teststation event.

On reception of an event, the event is written to a complex-specific event_log file at /spp/data/<COMPLEX_NAME>/event_log. When the `event_log` reaches the maximum size (approximately one Mbyte), the `event_logger` compresses the `event_log`, then truncates the file and continue logging events.

Other programs can request that events be sent via `rpc` to them. These programs can use the `libevent_client` library to establish a service and notify the `event_logger` what events it would like to see with a fair degree of simplicity. On reception of an event, once the log file is written, the linked list of interested programs is searched to see if the event matches the criterion requested. If there is a match, the event is sent to that program.

event_logger should never terminate, but must be killed. If a second copy of event_logger is started it attempts to kill the existing copy of the event_logger. There should only be one copy of event_logger running at any one time.

The following return code indicates a fatal error occurred.

```
-1   unknown option
```

log_event

log_event logs its STDIN to the event log as a single event.

log_event has the following format:

```
log_event [-c] [event number] -n NODE_ID
```

where:

- [event number]—Specifies is the event code to use in one of three ways:
  - Command line
  - First line of the input
  - Default
- [-c]—Specifies that event is displayed to the console in addition to logging it to the event_log
- -n NODE_ID—Specifies the node this event is being logged against

The default event code is used if one is not specified using one of the other methods. The command line method is used by specifying a decimal or hex (leading 0x) as the only number on the command line (the -c option optionally may be present). The input mode is used if a number is the first thing on the first line read from STDIN. The input mode overrides the command line. In the last case, the entire first line is not logged.

When entering text for an event, you may terminate and send the event with a ctrl-D. You may cancel the event with ctrl-C.

log_event always returns 0.

log_event is used by scripts such as the interrogators and extractors to put information into the event log as follows:

```
log_event [-c] [number] -n NODE_ID
```

The `-c` option displays event information output on the console as well. If the event severity is high enough, this happens automatically. `event_logger` displays any events that have a severity greater than the warning level.

The following two examples show how `log_event` can be used:

```
cat data_file | log_event 0x86340001 -n 0
```

This example puts an event in the event log with the event code of 0x86340001. The data will be the information contained in the file data_file.

```
echo “This is a test event” | log_event -n 0
```

This example puts the message “This is a test event” in the event log with the default event code from `log_event`.

Miscellaneous tools

The following miscellaneous tools are described in this section:

- `kill_by_name`
- `fix_boot_sector`

kill_by_name

The `kill_by_name` script kills processes by name rather than by process identification. The following is the usage of this script:

kill_by_name <file name> <signal to send> <process id to not kill>

Table 85 describes the options in `kill_by_name`.

Table 85 `kill_by_name` options

Option	Description
`file name`	Process name to kill.
`signal to send`	Default is kill command.
`process id to not kill`	Kills all processes that match the specified name except the one matching this ID.

If the third argument is used, the second argument must also be specified. For example:

```
kill_by_name foo 15 1234
```

fix_boot_vector

This `sppdsh` script restores the four words at the beginning of NVRAM to point to POST. These four words are used by the ENTRY firmware to determine which process was executing last when an HPMC, TOC, or reset occurs.

12 Scan tools

This chapter details most of the scan tools which include:

- `sppdsh`
- `do_reset`
- `jf-node_info`
- `jf-ccmd_info`
- `jf-reserve_info`

sppdsh

sppdsh is an enhanced version of the Korn Shell (ksh) with all of the functionality of ksh, as well as new commands that are suited to a diagnostic environment. sppdsh resides on the teststation in /spp/bin/sppdsh.

The diagnostic shell runs on a teststation that is totally independent of the system itself. The shell requires information about the complexes and nodes attached to the teststation. ccmd interrogates the complexes and nodes on the DART bus and generates a database of information on the teststation; it does not act unilaterally.

POST passes system information to ccmd through NVRAM about the system itself. If POST fails to initialize the system, ccmd will time out and print a warning. If this occurs, many diagnostic shell operations will not work as expected.

On start-up, the diagnostic shell reads the database that ccmd provides. If major changes are made to the system, sppdsh should be restarted to be sure that the shell has an accurate representation of the system. If ccmd is restarted, then the shell *must* be restarted.

sppdsh commands are sorted into the following five categories:

- Miscellaneous commands—Control the system behavior and aid in generating useful scripts
- Data transfer commands—Allow the user to transfer register state or memory information back and forth between the system and the teststation
- Data conversion commands—Reformat data to make it more useful
- System information commands—Provide information about the system hardware to run diagnostic upon
- I/O buffering commands—Aid in the testing of I/O devices and memory.
- Configuration commands—Alters a system configuration for POST after a reboot.
- SPP enhancements—New sppdsh commands for the V2500 server.

The commands in each category are described in the following sections.

Definitions

The following definitions will help user with the operation of `sppdsh`:

- node id—An identification (ID) that can be either the node IP name or a node number. To distinguish between one node number and another, the environmental variable, COMPLEX_NAME, indicates the complex. No complex can have non-unique node numbers.
- complex name—Identifies a grouping of nodes. The `ts_config` utility groups nodes into complexes where each node shares the same OS and memory space.
- all—A reference to all nodes associated with the only complex on the diagnostic bus in a single complex configuration.
- <n<node number> | node id>:<ring>:<path>:<part>:<field>—The general description of a register or group of registers that are accessible through scan. Each register is identified by its node, the scan ring that can access it, the part or device that contains it, the scan path within the part, and the text-based description of the register.
- address—A 40-bit value that allows access to the memory, CSR space, IO space and Core Logic bus space across all possible nodes in a complex.
- byte_size—The byte_size argument represents the number of bytes to access at an address. Valid byte_sizes are 1, 2, 4 or 8 but may also be limited by the type of memory accessed. If the byte_size argument is not used, the maximum valid size for the argument is used by default.
- value—A representation of the data transferred to a memory address or scan field.
- parameter—A name that represents configuration data that is initialized by POST. Parameters may be changed to aid testing or to deconfigure hardware that is marginal. Table 86 provides a list of valid parameters.

Table 86 sppdsh parameters

Parameter	Value
Unknown	0xff
Reserved	0x00
Pass	0x01
Fail	0x10
Deconfigured by POST	0x20
Empty	0x30
Deconfigured by software	0x40 [a]
16MB deconfigured	0x04
16MB 88-bit deconfigured to 80	0x24
16MB 88-bit deconfigured	0x34
16MB SW deconfigured	0x44
16MB 88-bit SW deconfigured to 80	0x64
16MB 88-bit SW deconfigured	0x74
64MB deconfigured	0x08
64MB 88-bit deconfigured to 80	0x28
64MB 88-bit deconfigured	0x38
64MB SW deconfigured	0x48
64MB 88-bit SW deconfigured to 80	0x68
64MB 88-bit SW deconfigured	0x78
128MB deconfigured	0x0c
128MB 88-bit v to 80	0x2c
128MB 88-bit deconfigured	0x3c
128MB SW deconfigured	0x4c

Parameter	Value
128MB 88-bit SW deconfigured to 80	0x6c
128MB 88-bit SW deconfigured	0x7c
64MB deconfigured to 16MB	0x89
64MB deconfigured to 16MB (88-bit to 80)	0xa9
64MB deconfigured to 16MB (88-bit)	0xb9
SW deconfigured 64MB to 16MB	0xc9
SW deconfigured 64MB to 16MB (88-bit to 80)	0xe9
SW deconfigured 64MB to 16MB (88-bit)	0xf9
128MB deconfigured to 16MB	0x8d
128MB deconfigured to 16MB (88-bit to 80)	0xad
128MB deconfigured to 16MB (88-bit)	0xbd
SW deconfigured 128MB to 16MB	0xcd
SW deconfigured 128MB to 16MB (88-bit to 80)	0xed
SW deconfigured 128MB to 16MB (88-bit)	0xfd
128MB deconfigured to 64MB	0x8e
128MB deconfigured to 64MB (88-bit to 80)	0xae
128MB deconfigured to 64MB (88-bit)	0xbe
SW deconfigured 128MB to 64MB	0xce
SW deconfigured 128MB to 64MB (88-bit to 80)	0xee
SW deconfigured 128MB to 64MB (88-bit)	0xfe

a. System memory can be modified through partial deconfiguration.

- buf[1..4]—A buffer is a 4K byte block of memory on the test station that is used as a temporary holding area.

- backplane_serial_number—Identifies a specific board on the diagnostic network. This number may be read with the COP command. It is used to assign new node numbers or complex serial numbers.
- complex_serial_number—Identifies all the nodes in a complex. Software licensing is often based on the complex serial number.
- key—A 32-bit hexadecimal number used as an encryption code for complex serial numbers.
- cop_id—A name associated with a board in a node. Table 87 lists valid cop IDs.

Table 87 Valid COP IDs

ID	Description
scub	A system communications and utility board
mib	A midplane or backplane
pb0l	A processor board on the left side of the cabinet
pb0r	A processor board on the right side of the cabinet
pb1l	A processor board on the left side of the cabinet
pb1r	A processor board on the right side of the cabinet
pb2l	A processor board on the left side of the cabinet
b2r	A processor board on the right side of the cabinet
pb3l	A processor board on the left side of the cabinet
pb3r	A processor board on the right side of the cabinet
pb4l	A processor board on the left side of the cabinet
pb4r	A processor board on the right side of the cabinet
pb5l	A processor board on the left side of the cabinet
pb5r	A processor board on the right side of the cabinet
pb6l	A processor board on the left side of the cabinet
pb6r	A processor board on the right side of the cabinet

ID	Description
pb7l	A processor board on the left side of the cabinet
pb7r	A processor board on the right side of the cabinet
mb0l	A memory board on the left side of the cabinet
mb1l	A memory board on the left side of the cabinet
mb2r	A memory board on the right side of the cabinet
mb3r	A memory board on the right side of the cabinet
mb4l	A memory board on the left side of the cabinet
mb5l	A memory board on the left side of the cabinet
mb6r	A memory board on the right side of the cabinet
mb7r	A memory board on the right side of the cabinet
iolf	An IO board on the left-front
iolr	An IO board on the left-rear
iorf	An IO board on the right-front
iorr	An IO board on the right-rear

- Device_name—Refers to a major electrical component or subsection of a node. Examples of device names are:
  - SPAC—Processor agent chip
  - SMAC—Memory chip
  - STAC—SCI transfer chip
  - SAGA—IO controller chip
  - ERAC—Crossbar network chip
  - CPU—Processor
- ID_number—Refers to a specific instance of the device named.
- <A|C|M|D|I|S>—Notation that refers to the processor agent chip, processor, memory, DIMM, IO chip or SCI chip.

- memory size—An argument used to deconfigure larger amounts of memory across all memory boards on a node.
- net cache size—Refers to the memory shared between nodes in each node's network cache. The network cache should be the same across all nodes in a complex.

Miscellaneous commands

`sppdsh` miscellaneous commands are described below:

- `assert <node_id>`—Assert reset on node_id; a deassert must follow.
- `assert_soft <node id>`—Asserts a soft reset on node id.
- `assert_toc <node_id> alt_name <E_name> <ID_number>`—Asserts transfer of control on `node_id`.
- `deassert <node_id>`—Negate reset to `<node_id>`.
- `clock <stop|clock1> <node>:<ring>:<part>`—Issue special clock operations to node:ring:part.
- `fprint "hello world %s is %d" $variable 0xff`—Format output.
- `alt_name <E_name> <ID_number>`—Return the alternate name of a system board or component. For example, entering **`alt_name EPIC 0`**, produces `IOLF_B`. This command does not support the EPUC or EMUC.
- `deassert <node id>`—Deassert reset to node id.
- `reboot <node id| complex name| all> <default|tc_stand|tc_int|obp|epsdv|post_int|loader>`—Reboots the node or complex specified. It can use default or new values for POST configuration data. Default values are determined by POST after ignoring existing values. After POST runs, control can be transferred to OBP, EPSDV, POST interactive mode, Test Controller in stand alone mode, or the Test Controller in interactive mode. When default is specified control is transferred to OBP.
- `clock <node id> [ext|nom|high]`—Changes the clock margin on all nodes in contact with the teststation.

- `power <node id> supply[1..4] [low|nom|up]`—Changes the power margin on the supply indicated across all nodes in contact with the test station.
- `power <node id> supply[1..4] [low|nom|up`—Changes the power margin on the supply indicated across all nodes in contact with the test station.
- `pswitch <node id>`—Identifies whether N or N+1 fans have been enabled for the system. This switch is located on the SCUB board of a node.
- `pce <node id|complex name|all> [-c <n|u|e>] [-r <on|off>] [-p <l|n|u>]`—Displays the current power, clock and temperature state where:
  - `-c [n|u|e]`—Sets the following clock tolerances on the current node:
    - `n`[ominal]—Nominal frequency
    - `u`[pper]—Upper frequency
    - `e`[xternal]—External connector
  - `-r`—Sets the power flag to on or off
  - `-p [l|n|u]`—Displays the following power supply voltages tolerances on the current nodes.
    - `l`[ower]—Lower voltage tolerance
    - `n`[ominal]—Nominal voltage
    - `u`[pper]—Upper voltage tolerance

NOTE Clocks are stopped by putting all scannable parts in internal scan mode. Other scan paths are not allowed when clocks are stopped. A system reset must follow an internal scan node operation.

Data transfer commands

This section lists the `sppdsh` data transfer commands. The addresses in the data transfer commands are 40 bits. Underbars are ignored in addresses.

NOTE For clarity, a 0x0 style notation is returned by the shell rather than the 16#0 notation of ksh. The 16#0 notation is acceptable for data that can be expressed in 32 bits or less.

- `list <n<node number> | node id>:<ring>:<path>:<part>:<field>`—Lists the possible paths, parts or fields that match the argument. Common wild card symbols are supported by this command to help identify fields names.
- `put [<node_number>:]<address>[,<byte_size>] <value>`—Starting at the node indicated by `<node id>`, write `<byte_size>` bytes into the memory address using the `<value>`. `sppdsh` is aware of the various memory sizes assumed at various addresses and retrieves the appropriate size (For example, 0x20:0x21 = 0x30)
- `put [-q] n<node_number>:<ring>:<path>:<part>:<field>`—Writes the scan location node `<node_id>:<ring>:<path>:<part>:<field>`. The `-q` option is used to display the result without the scan field name. If a node number is used as the node id then an “n” should precede the node number as “n0”.
- `get [<node_number>:]<address>[,<byte_size>] [<iterations>]`—Reads `<byte_size>` bytes from the memory location `<address>` on node `<node id>`. This command can be repeated with the address incremented. One or `<iterations>` different addresses will be read.
- `get [-q] n<node_number>:<ring>:<path>:<part>:<field>`—Reads scan location node `<node_id>:<ring>:<path>:<part>:<field>`. The `-q` option is used then the result should be displayed without the scan field name. The `-a` option displays both the address and the data. The `-b` option eliminates leading zeros. If a node number is used as the node id then an “n” should precede the node number as “n0”
- `block n<node_number>:<ring>:<path>`—Reads the scan ring at `<node id>:<ring>:<path>` and lock the scan ring image for `bget` and `bput` operations.
- `bget [-q] <part>:<field>`—Extracts data from the locked scan ring image. When the `-q` option is used, the results are displayed without the scan field name

- `bput [-q] <part>:<field> <value>`—Inserts data into the locked scan ring image. When the `-q` option is used, the results are displayed without the scan field name.
- `bunlock n<node_number>:<ring>:<path>`—Writes the scan ring image and unlocks it.
- `packet [-q] [NR | R=number] [P=number] [6=number] node8_0 <packet symbols>`—Requests input to a xbar device from SPAC 0 on node 8. The request waits for a response and returns it. The `-q` option suppresses some output. Other arguments are as follows:
  - NR—No response
  - R = N—Response of N symbols
  - P = N—Select port N
  - 6 = N—Use N as the R6 symbol

  The following is an example of this command:

  ```
  packet R=3 P=1 node0_2 rd_short R=3 lgth=1 route=4
  adr=0 dl=0 lcl=1 mstr=2a tid=1c size=3 Q=0
  ```
- `file_to_mem <file_name> <address>`—Reads a file and loads the file into memory starting at address.
- `mem_cmp <address1> <address | buffer> <size>`—Compares the memory at `address1` to `address1+size` to that at `address2`. `size` is a value in bytes.
- `mem_dump <address> [size]`—Dumps the memory starting at `address`. `size` is a value in 64-bit words.
- `mem_cpy <address | buffer> <address | buffer> [size]`—Copies the memory from `address1` to `buffer1` - up to `size` or 4K bytes. `size` is a value in 64-bit words.
- `tag_dump <address> [size]`—Dumps the tags associated with the cache line of `address` and repeat for `size` cache lines.
- `tag_cpy <address> <data> [size]`—Copies the data into the tags associated with the cache line of `address` and repeat for `size` cache lines.
- `ecc_dump <address> [size]`—Dumps the ECC bits associated with the cache line of `address` and repeat for `size` cache lines.

- `ecc_cpy <address> <data> [size]`—Copies the data into the ECC associated with the cache line of `address` and repeats for `size` cache lines.

Data conversion commands

Data conversion commands manipulate, evaluate or interpret data within the diagnostic shell. They support a variety of logical, arithmetic and string based operations. The following example is representative of the data conversion commands:

```
abc='and 0xff 0x55'
```

Unless otherwise stated, these commands support data types that are greater than 32 bits, not supported under `ksh`.

The following is a list of `sppdsh` data conversion commands:

`and <arg1> <arg2>`—And two data arguments.

or <arg1> <arg2>—OR two data arguments. For example:

```
abc=`or 0xff 0x55`
```

`xor <arg1> <arg2>`—Exclusively OR two data arguments. For example:

```
abc=`xor 0xff 0x55`
```

`even_parity <arg1> <arg2>`—Return even parity. For example:
abc=`even_parity 0xff``

`odd_parity <arg1> <arg2>`—Return odd parity. Parity is based on 4 bytes, as an example:

```
abc=`odd_parity 0xff`
```

`comp <arg1> <arg2>`—Compare two data arguments. For example:

```
abc=`comp 0xff 0xff`
```

`rshift <arg1> <arg2>`—Right shift two data arguments. For example:

```
abc=`rshift 0x55 0x1`
```

`lshift <arg1> <arg2>`—Left shift two data arguments. For example:

```
abc=`lshift 0x55 0x1`
```

`l_add <arg1> <arg2>`—Left add two data arguments. For example:

```
abc=`l_add 0x55 0x1`
```

`l_sub <arg1> <arg2>`—Left subtract two data arguments. For example:

```
abc=`l_sub 0x55 0x1`
```

`l_mod <arg1> <arg2>`—Left modulo two data arguments. For example:

```
abc=`l_mod 0x55 0x1`
```

`l_mult <arg1> <arg2>`—Left multiply two data arguments. For example:

```
abc=`l_mult 0x55 0x1`
```

`b2h <arg1> <arg2>`—Convert a binary number to hex (abc = 0xb). This command is limited to 32-bit data types. For example:

```
abc=`b2h 1011`
```

`h2b <arg1> <arg2>`—Convert a hex number to binary (abc = 1011). This command is limited to 32-bit data types. For example:

```
abc=`h2b 0xb`
```

`conv <arg1> <arg2>`—Converts from hex to decimal. This command is limited to 32-bit data types. For example:

```
abc=`conv 0xb`
```

`conv <arg1> <arg2>`—Convert from decimal to hex. For example:

```
abc=`conv 11`
```

converts 11 from decimal to hex and assign it to abc (abc = 0xb). This command is limited to 32-bit data types.

`s_tos <arg1> <arg2>`—Removes underbar from a hex number. For example:

```
abc=`s_tos 0xff_abcd`
```

Converts 0xff_abcd to 0xffabcd and assigns it to abc.

System information commands

The following are system information commands. For all these command, `-v` produces the verbose manufacturing name, `-q` produces the name alone without additional information, and `-a` produces the available memory.

`complex <c_name>`—Set the current, default complex to be complex c_name. If only one complex is available, this command is not necessary.

`node <node _number>`— set default node to be node _number in the current complex.

`fi_node`—Find all available nodes in the current complex.

`fi_cpu [-v] [-q] <node_number>`—Find all available processors of `node_number` in the current complex.

`fi_emb [-v] [-q] <node_number>`—Find all available EMBs of `node_number` in the complex.

`fi_sci [-v] [-q] <node_number>`—Find all available SCIs of `node_number` in the current complex.

`fi_pic [-v] [-q] <node_number>`—Find all available EPICs of `node_number` in the current complex.

`fi_pac [-v] [-q] <node_number>`—Find all available EPACs of `node_number` in the current complex.

`fi_rac [-v] [-q] <node_number>`—Find all available ERACs of `node_number` in the current complex.

`fi_slice [-v] [-q] <node_number>`—Find all available slices of `node_number` in the current complex.

`fi_mem_inst [-a] [-q] <node_number>`—Find the installed memory size per EMB of `node_number` in the current complex.

Configuration commands

The following is a list of the `sppdsh` configuration commands:

`retrieve <node_number>`—Retrive the `node_number` configuration parameters to the teststation from NVRAM.

`replace <node_number>`—replaces the `node_number` configuration parameters from the teststation to NVRAM.

`cput <node_number> parameter_name 0xnnnnnnnn`–Set the configuration `parameter_name` of `node_number` to 0xnnnnnnn in the teststation buffer.

`cget <node_number> parameter_name`—Get the value stored in the configuration `parameter_name` of `node_number` in the teststation buffer.

I/O buffering commands

This section presents a list of the sppdsh I/O buffering commands. For these commands, four default buffers are created: buf1 - buf4.

buf_cmp buf1 buf2—Compares two buffers. Null is returned if they are the same. If they are different, the index and data of the first conflict is returned.

buf_cpy buf1 buf2—Copy buf1 to buf2

buf_clear buf—Clear buf1

seed [get|set 0xseed_value]—Set or get a seed value.

buf_mod [-bw|-s len value] buf_name [value | key_data] [nbr] [offset]—Write to buffer. The following are three examples:

1. **buf_mod buf1 0x01234567 10 2**

   writes 0x01234567 10 times with an offset of 2 words.

2. **buf_mod -b buf1 0x3d 1 10**

   writes the byte 0x3d once at 0x10

3. **buf_mod -s 5 0 buf2**

   write five zeros then five ones for all of the buffer space using the following key data:

   - rand—Produces random data based on the seed
   - zeroes—Produces all zero data
   - ones—Produces all one data
   - alt1—Alternates zeros and ones
   - alt2—Alternates ones and zeros

buf_print buf1—Print buffer contents.

buf_read buf1 [size]—Prints the value of a byte in the given buffer.

Memory transfer commands

The following is list of all sppdsh memory transfer commands:

file_to_mem file_name address—Reads and loads a file into memory starting at address.

`mem_cmp addr1 addr2 size`—Compares the memory at `addr1` to (`addr1+size`) to that at `addr2`.

`mem_cmp addr1 buf1 size`—Compares the memory at `addr1` to (`addr1+size`) to that at `buf1`.

`mem_dump addr [size]`—Dumps the memory starting at `addr`.

`mem_cpy addr1 buf1 [size]`—Copies the memory from `addr1` to `buf1` up to `size` or four Kbytes.

`mem_cpy buf1 addr1 [size]`—Copies the memory from `buf1` to `addr1` up to `size` or four Kbytes.

`tag_dump <address> [size]`—Dump the tags associated with the cache line of `<address>`. Repeat for `[size]` cache lines.

`tag_cpy <address> <data> [size]`—Copy the data into the tags associated with the cache line of `<address>`. Repeat for `[size]` cache lines.

`ecc_dump <address> [size]`—Dump the ecc bits associated with the cache line of `<address>`. Repeat for `[size]` cache lines.

`ecc_cpy <address> <data> [size]`—Copy the data into the ecc associated with the cache line of `<address>`. Repeat for `[size]` cache lines.

Map of alternate names

Table 88 lists the alternate names of ring numbers to system parts.

Table 88 **System rings to alternates names**

Ring	Parts	Alternate names
0	pb0l, p0l, pb0r	[pcxu], spac0, [pcxu]
1	pb1r, p1l, pb1l	[pcxu], spac1, [pcxu]
2	pb2l, p2l, pb2r	[pcxu], spac2, [pcxu]
3	pb3r, p3l, pb3l	[pcxu], spac3, [pcxu]
4	pb4l, p4l, pb4r	[pcxu], spac4, [pcxu]
5	pb5r, p5l, pb5l	[pcxu], spac5, [pcxu]

Ring	Parts	Alternate names
6	pb6l, p6l, pb6r	[pcxu], spac6, [pcxu]
7	pb7r, p7l, pb7l	[pcxu], spac7, [pcxu]
8	mb0l_m, mb0l_t	smac0, [stac0]
9	mb1l_m, mb1l_t	smac1, [stac1]
10	mb2r_m, mb2r_t	smac2, [stac2]
11	mb3r_m, mb3r_t	smac3, [stac3]
12	mb4l_m, mb4l_t	smac4, [stac4]
13	mb5l_m, mb5l_t	smac5, [stac5]
14	mb6r_m, mb6r_t	smac6, [stac6]
15	mb7r_m, mb7r_t	smac7, [stac7]
16	iolf_b, iolf_a	saga0, saga4
17	iolr_b, iolr_a	saga1, saga5
18	iorr_b, iorr_a	saga2, saga6
19	iorf_b, iorf_a	saga3, saga7
20	rol, r2r	srac0, srac1
21	r1l, r3r	srac2, srac3
22	u_p, u_m	spuc, smuc

do_reset

`do_reset` performs one of four levels of reset on a node or complex. The first argument is either a node ID, complex, or the keyword, `all`, which resets all nodes. If no nodes are specified, the default is to reset all nodes in contact with the teststation. If a node number is specified, the level argument must be specified as well.

The second argument specified is the level of reset. All levels of reset are expressed as numbers. If no level is specified then a “reset” level of 1 is assumed.

The following reset levels are available:

1. JTAG controller SCUB reset, hard reset, clear options bits, and send messages to ccmd
2. JTAG controller SCUB reset and system soft reset
3. JTAG controller SCUB reset
4. TOC reset

`do_reset` halts any scan activity taking place on the nodes. Larger systems require more time to reset. Reset continues after this command returns.

If the reset completes normally, `do_reset` returns zero. If the operation cannot be completed, `do_reset` returns a nonzero exit code.

If clocks are not stopped, then `do_reset` may also change the boot option after the reset. The boot options are as follows:

- `OBP`
- `post_interactive`
- `spsdv`
- `tc_standalone`
- `tc_interactive`
- `loader`

jf-node_info

`jf-node_info` displays the IP address, UDP port and JTAG firmware version string for each node in a complex. The `-e` option adds the ethernet address to the display. The `-c` option adds the core version to the display.

jf-ccmd_info

`jf-ccmd_info` displays information about active V2500 nodes connected to the diagnostic LAN. It has the following format:

```
jf-ccmd_info
```

The display includes the Ethernet address, IP address, Complex Serial number, Node number, environmental LED status, and the Diagnostic node name of each active V2500 node.

`jf-ccmd_info` sends a broadcast packet to all nodes on the diagnostic LAN requesting this information. `jf-ccmd_info` accumulates responses received within a short timeout period then sorts the responses based on node name.

The JTAG utility firmware responds to the request with output similar to the following:

```
joker-t(hw2a):/users/sppuser$ jf-ccmd_info

                              Complex  Node Env  Pwr  Cub  Diagnostic
Ethernet Addr  IP Address     Serial #   #  Led  Sts  Sts  Node name
-------------- -------------- ---------- -  ---  ---  ---  ----------
0x00A0D900BF03 15.99.111.116  SN12757550 0  0x00 0x80 0x00 hw2a-0000
0x00A0D900C3A3 15.99.111.117  SN13169380 0  0x00 0x80 0x00 hw2b-0000
```

CAUTION

`jf-ccmd_info` displays information about all active V2500 nodes that answer the broadcast request, even if the node is not configured.

If the `jf-ccmd_info` utility displays information about a node, but the node has not been detected by the `ccmd` daemon, then the node is not configured. Use the `ts_config` utility to configure the node.

jf-reserve_info

Before using the JTAG scan interface on the Utilities board, teststation utilities must *reserve* the JTAG hardware on a time-sharing basis. It has the following format:

`jf-reserve_info`

`jf-reserve_info` sends a broadcast packet to all nodes on the diagnostic LAN requesting the latest JTAG reservation information. The JTAG utility firmware responds to the request with output similar to the following:

```
joker-t(hw2a)% jf-reserve_info

RSV Node Node name        UID     PID  TTY     Time of reserve     Command
--- ---- -------------  ------- ----- -----   ------------------ -------
 -   0   hw2a-0000       sppuser  2934 pts/3   Oct 26 16:40:19:229 sppdsh
```

The RSV column indicates whether the JTAG hardware is currently reserved. This column may contain a Y (indicating YES) or “-], indicating no current reservation.

If the JTAG hardware is reserved, the output includes information about the teststation utility that is currently using the JTAG hardware.

If the JTAG hardware is not reserved, the process information shown is historical data about the last process that reserved the JTAG hardware on the specified node.

A List of diagnostics

This appendix provides a list of all utilities and diagnostics in this book and where they are located.

Table 89 List of diagnostics

Name	Locations
address_decode	Page 214
arrm	Page 215
autoreset	Page 50
ccmd	Page 40
consolebar	Page 219
cpu3000	Chapter 7, page 125
cpu3000_decode	Page 131
cxtest	Chapter 5, page 101
dcm	Page 53
dfdutil	Page 224
diag_version	Page 261
do_reset	Page 284
est	Chapter 10, page 183
est_config	Page 50
event_logger	Page 263
fix_boot_vector	Page 266
flash_info	Page 261
fwcp	Page 235
fw_init	Page 236
get_node_info	Page 238

Name	Locations
hard_logger	Page 240
io3000	Chapter 8, page 133
io_tr	Page 164
jf-ccmd_info	Page 286
jf-node_info	Page 285
jf-reserve_info	Page 287
kill_by_name	Page 266
lcd	Page 242
load_eprom	Page 243
log_event	Page 264
mem3000	Chapter 9, page 165
pdcfl	Chapter 6, page 119
pim_dumper	Page 246
POST	Chapter 3, page 53
rdr_dumper.fw	Page 234
rdr_formatter	Page 234
set_complex	Page 248
soft_decode	Page 250
scan_sram	Page 234
sppdsh	Page 268
spp_console	Page 251
tc_init	Page 255
tc_ioutil	Page 257
tc_show_struct	Page 258

Name	Locations
ts_config	Page 23
ver	Page 262
xconfig	Page 42
xsecure	Page 51

Index